شبکه های کامپیوتری

نویسنده

اندرواس.تنن باوم

فهرست مطالب

مقدمه

هر یک از سه قرن گذشته را با یک تکنولوژی خاص بعنوان نماد آن قرن می‌شناسیم. قرن هیجدهم عصر سیستمهای بزرگ مکانیکی و انقلاب صنعتی بود، و قرن نوزدهم عصر بخار. تکنولوژی کلیدی قرن بیستم نیز جمع‌آوری، پردازش و توزیع اطلاعات بود. شبکه‌های گسترده و بین‌المللی تلفن، اختراع رادیو و تلویزیون، تولد و گسترش باورنکردنی صنعت کامپیوتر، و پرتاب ماهواره‌های مخابراتی از نمادهای این عصر هستند.

با رشد سریع تکنولوژیهای جمع‌آوری، پردازش و توزیع اطلاعات، این زمینه‌ها بسرعت در هم ادغام شده، و تفاوتهای آنها در حال محو شدن است. شرکتهایی که در اقصی نقاط دنیا شعبه و نمایندگی دارند، می‌توانند فقط با فشار یک دکمه از آخرین وضعیت دفاتر خود (حتی دور افتاده‌ترین آنها) مطلع شوند. اما جالب اینجاست که رشد تقاضا برای روشهای پیشرفته‌تر پردازش اطلاعات همیشه یک گام از سرعت رشد این تکنولوژیها جلوتر است.

با اینکه صنعت کامپیوتر از صنایع دیگر (از جمله صنایع اتومبیل، و حمل و نقل هوایی) نسبتاً جوانتر است، اما در مدتی بس کوتاه به پیشرفتهای چشمگیری دست یافته است. در دو دهۀ اول، سیستمهای کامپیوتری بسیار متمرکز بودند، و معمولاً در یک اتاق بزرگ جا می‌گرفتند. کم نبودند مراکزی که این اتاقها دیوارهای شیشه‌ای داشتند، و بازدیدکنندگان با حیرت این موجودات عجیب‌الخلقۀ الکترونیکی را برانداز می‌کردند. دانشگاهها و شرکتهای متوسط معمولاً یکی دو کامپیوتر بیشتر نداشتند، و تعداد شرکتهایی که استطاعت خرید بیش از یک دوجین از آنها را داشته باشند، چندان زیاد نبود. هیچکس (شاید غیر از نویسندگان داستانهای علمی-تخیلی) حتی نمی‌توانست تصور کند که تا قبل از پایان قرن بیستم کامپیوترهایی با همان قدرت را بتوان روی یک تمبر پستی جای داد، و آنها را بصورت انبوه تولید کرد.

پیوند فرخندۀ کامپیوتر و مخابرات اتفاقی بود که هر دو صنعت را دچار تحولات عظیم کرد. اکنون دیگر مفهوم اتاقی با یک کامپیوتر بزرگ بنام «مرکز کامپیوتر»، که افراد کارهایشان را به آنجا می‌آورند، بکلی منسوخ شده است. مدل قدیمی کامپیوتر بزرگی که تمام کارهای محاسباتی سازمان را انجام می‌دهد، اکنون جای خود را به تعداد زیادی کامپیوتر کوچک متصل به هم داده است. به این سیستمها **شبکه‌های کامپیوتری** (computer networks) گفته می‌شود؛ موضوع این کتاب نیز طراحی و ساختار این شبکه‌هاست.

در این کتاب هر جا از «شبکۀ کامپیوتری» سخن می‌گوئیم، منظورمان مجموعه‌ای از کامپیوترهای مستقل است، که با یک تکنولوژی واحد به هم متصل شده‌اند. دو کامپیوتر وقتی «به هم متصلند»، که بتوانند با یکدیگر اطلاعات رد و بدل کنند. الزامی نیست که این اتصال از طریق سیمهای مسی باشد؛ فیبرهای نوری، امواج مایکروویو و مادون قرمز، و ماهواره‌های مخابراتی هم می‌توانند عامل این ارتباط باشند. بعداً خواهید دید که اندازه، شکل و ساختار

شبکه‌ها می‌تواند بسیار متفاوت باشد. همچنین بسیاری از افراد وقتی می‌شنوند که اینترنت یا وب هیچکدام شبکهٔ کامپیوتری نیستند متعجب می‌شوند؛ اما در پایان این کتاب علت آنرا هم خواهید فهمید. فعلاً همین قدر کافیست بدانید که: اینترنت یک شبکه نیست، بلکه شبکه‌ایست از شبکه‌ها، و وب نیز یک سیستم توزیع شده است که بر پایهٔ اینترنت کار می‌کند.

لازم است همین جا به یک اشتباه رایج بین دو اصطلاح **شبکهٔ کامپیوتری** و **سیستم توزیع شده** (distributed system) اشاره کنم. یک سیستم توزیع شده مجموعه‌ایست از چندین کامپیوتر مستقل، که کاربر آنرا به شکل یک سیستم واحد و متجانس می‌بیند. در این سیستمها معمولاً یک لایهٔ نرم‌افزاری (روی سیستم عامل) بنام **میان‌افزار** (middleware) است، که مدل موردنظر را پیاده‌سازی می‌کند. **وب** (World Wide Web) نمونه‌ای از یک سیستم توزیع شده است، که در آن همه چیز از دیدگاه کاربر یک سند (صفحهٔ وب) بنظر می‌رسد.

در شبکهٔ کامپیوتری این تجانس، مدل و نرم‌افزار وجود ندارد. کاربران بطور مستقیم با کامپیوترها در تماسند، و هیچ کوششی برای ایجاد تجانس بین آنها صورت نمی‌گیرد. کاربر بروشنی تفاوتهای نرم‌افزاری و سخت‌افزاری کامپیوترها را می‌بیند، و اگر بخواهد برنامه‌ای را روی یکی از کامپیوترها اجرا کند، باید ابتدا وارد آن شود (log on).

در حقیقت، یک سیستم توزیع شده نرم‌افزاریست که روی شبکه کار می‌کند، و تجانس و شفافیت آن توسط این نرم‌افزار تأمین می‌شود. بهمین دلیل تفاوت سیستم توزیع شده با یک شبکه بیشتر در نرم‌افزار (بویژه سیستم عامل) نهفته است تا سخت‌افزار.

با این همه، شباهتهای زیادی نیز بین این دو وجود دارد. مثلاً، سیستمهای توزیع شده و شبکه‌ها هر دو به انتقال فایل نیاز دارند؛ تفاوت در اینست که این کار را چه کسی انجام می‌دهد، سیستم یا کاربر. با آنکه این کتاب دربارهٔ شبکه‌های کامپیوتری است، بسیاری از مطالب آن در سیستمهای توزیع شده نیز مصداق دارد. برای کسب اطلاعات بیشتر دربارهٔ سیستمهای توزیع شده به (Tanenbaum and Van Steen, 2002) نگاه کنید.

۱-۱ کاربردهای شبکه‌های کامپیوتری

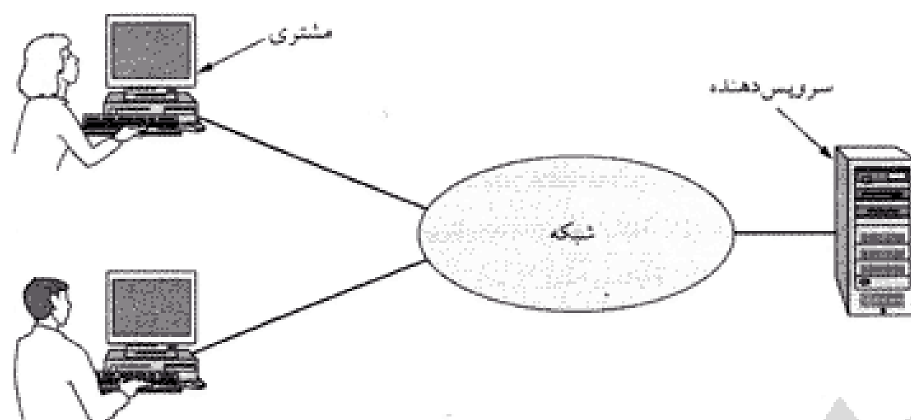
قبل از پرداختن به جزئیات فنی، بهتر است کمی دربارهٔ اینکه چرا مردم به شبکه‌های کامپیوتری اهمیت می‌دهند و چرا از آنها استفاده می‌کنند، صحبت کنیم (چراکه اگر کسی به شبکه اهمیت نمی‌داد، اصلاً شبکه‌ای ساخته نمی‌شد). ابتدا از کاربردهای سنتی (از قبیل شرکتها و افراد) شروع می‌کنیم، و سپس به کاربردهای جدیدتر (مانند شبکه‌های متحرک و خانگی) خواهیم پرداخت.

۱-۱-۱ کاربردهای تجاری

اکثر شرکتها تعداد زیادی کامپیوتر برای کارهای مختلف (تولید، انبارداری، فروش، و حسابداری) دارند. شاید در ابتدا این کامپیوترها از یکدیگر جدا باشند، ولی در مرحله‌ای از کار برای یکپارچه کردن اطلاعات کل شرکت، مدیریت تصمیم می‌گیرد تا آنها را به هم متصل کند.

به بیان کلی‌تر، **اشتراک منابع** (resource sharing) به ما اجازه می‌دهد تا برنامه‌ها، تجهیزات و بخصوص داده‌ها را (صرفنظر از موقعیت فیزیکی افراد و منابع) در اختیار همهٔ آنهایی که به این شبکه متصلند، قرار دهیم. ساده‌ترین مثال آن، چاپگریست که برای تمام کارکنان یک دفتر به اشتراک گذاشته شده است. پیداست که تک تک این افراد به یک چاپگر اختصاصی نیاز ندارند، و علاوه بر آن یک چاپگر شبکه اغلب ارزانتر، سریعتر و کم هزینه‌تر از تعداد زیادی چاپگرهای پراکنده است.

با این حال، اشتراک اطلاعات بسیار مهمتر از اشتراک تجهیزات فیزیکی (مانند چاپگر، اسکنر، و CD نویس) است. امروزه تمام شرکتهای بزرگ و متوسط (و بسیاری از شرکتهای کوچک) بشدت به اطلاعات کامپیوتری خود



شکل ۱-۱. شبکه‌ای با دو مشتری و یک سرویس‌دهنده.

(از قبیل اطلاعات مشتریان، انبار، سندهای مالی و حسابداری، و اطلاعات مالیاتی) وابسته‌اند. بانکی که تمام کامپیوترهای آن از کار افتاده باشند، پنج دقیقه هم نمی‌تواند دوام بیاورد. حتی شرکتهای کوچکی مانند آژانس‌های مسافرتی و دفاتر خدمات حقوقی نیز بشدت به اطلاعات کامپیوتری خود متکی هستند.

در یک شرکت کوچک تمام کامپیوترها به احتمال زیاد در یک دفتر (و یا حداکثر یک ساختمان) قرار دارند، در حالیکه کامپیوترهای یک شرکت بزرگ می‌تواند در یک شهر یا کشور (و حتی در قاره‌های مختلف) پراکنده باشد. در این حالت، ممکنست مدیر فروشی که در نیویورک نشسته، به موجودی انبار شرکت در سنگاپور نیاز داشته باشد. بعبارت دیگر، حتی ۱۵۰۰۰ کیلومتر فاصله هم نباید خللی در دسترسی به اطلاعات وارد کند. در واقع می‌توان گفت، ما بدنبال «از بین بردن فاصله‌ها» هستیم.

در ساده‌ترین شکل، اطلاعات شرکت می‌تواند در یک یا چند پایگاه داده متمرکز باشد، و کارمندان شرکت بایستی بتوانند از راه دور به آنها دسترسی داشته باشند. در این مدل، اطلاعات در کامپیوترهای پُرقدرتی بنام **سرویس‌دهنده** (server) - که اغلب در یک مرکز و تحت کنترل سرپرست سیستم قرار دارند - نگهداری می‌شوند. کارمندان نیز، که در اینجا به آنها**مشتری** (client) گفته می‌شود، از راه دور و از پای کامپیوترهای معمولی خود به این اطلاعات دسترسی پیدا می‌کنند. (گاهی به فردی که از کامپیوتر استفاده می‌کند، نیز «مشتری» گفته می‌شود؛ بهر حال، از فحوای متن باید بتوانید متوجه شوید که منظور کامپیوتر است یا کاربر.) اتصال کامپیوترهای سرویس‌دهنده و مشتری از طریق شبکه صورت می‌گیرد (شکل ۱-۱ را ببینید). در این شکل شبکه به صورت یک بیضی ساده نشان داده شده است؛ وقتی بخواهیم شبکه را بصورت کلی و انتزاعی (و بدون هیچگونه جزئیاتی) نشان دهیم، از این روش استفاده خواهیم کرد.

به این آرایش **مدل مشتری-سرویس‌دهنده** (client-server model) گفته می‌شود، و در بسیاری از شبکه‌های کوچک و بزرگ کاربرد دارد چون مستقل از فاصله است. وب نیز بر مبنای مدل مشتری-سرویس‌دهنده ساخته شده است؛ وقتی یک صفحهٔ وب را باز می‌کنید، در واقع آنرا از سرویس‌دهندهٔ وب دریافت کرده، و در کامپیوتر خود (که در اینجا مشتری است) نمایش می‌دهید. در اکثر مواقع یک سرویس‌دهنده می‌تواند به تعداد زیادی مشتری سرویس بدهد.

اگر مدل مشتری-سرویس‌دهنده را دقیقتر بررسی کنیم، متوجه می‌شویم که دو **پروسس** (process) در آن دخیل هستند: یک پروسس روی کامپیوتر مشتری، و دیگری روی کامپیوتر سرویس‌دهنده. ارتباط از لحظه‌ای آغاز می‌شود، که پروسس مشتری از طریق شبکه یک پیام به پروسس سرویس‌دهنده فرستاده، و سپس به انتظار پاسخ

آن می‌ماند. وقتی پروسس سرویس‌دهنده درخواست مشتری را دریافت کرد، کار خواسته شده را انجام می‌دهد (یا اطلاعات خواسته شده را تهیه می‌کند)، و پاسخ را به مشتری پس می‌فرستد. این فرآیند را در شکل ۱-۲ ملاحظه می‌کنید.

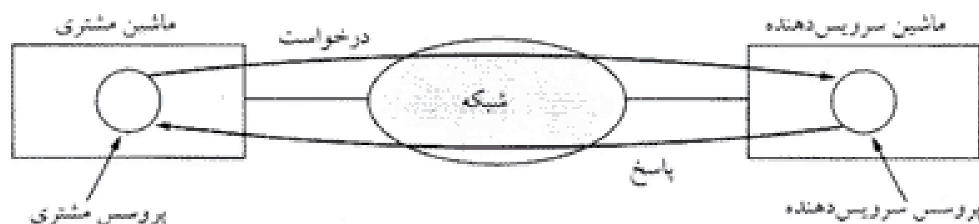
گاهی در یک شبکهٔ کامپیوتری رابطهٔ بین افراد اهمیت بیشتری نسبت به تبادل اطلاعات بین کامپیوترها دارد. چنین شبکه‌ای در واقع یک **رسانهٔ ارتباطی** (communication medium) است. امروزه دیگر تقریباً هیچ شرکتی را نمی‌توان یافت که از سرویس **پُست الکترونیک (ایمیل**: e-mail) استفاده نکند، و در واقع بسیاری از ارتباطات روزمرهٔ کارمندان از همین طریق صورت می‌گیرد. این روش آنقدر ساده و کارآمد است که خود باعث بروز مشکلات جدیدی شده است، چون رؤسای شرکتها هم یاد گرفته‌اند چطور فقط با فشار یک دکمه می‌توانند پیامهای (اغلب بی محتوای) خود را به این طرف و آن طرف بفرستند!

اما ایمیل تنها شکل از ارتباطات پیشرفته‌ای نیست که به لطف شبکه‌های کامپیوتری ممکن شده است. در یک شبکه، دو نفر که فاصلهٔ زیادی هم از یکدیگر دارند، می‌توانند بطور مشترک روی یک گزارش یا مقاله کار کنند. وقتی یکی از آنها تغییری در این گزارش می‌دهد، دیگری بلافاصله آنرا خواهد دید (و دیگر نیازی نیست روزها به انتظار پُستچی چشم به در بدوزد). با این روش دیگر نیازی نیست غصهٔ هماهنگ کردن کارمندانی که هر کدام ساز خود را می‌زنند، بخورید.

یکی دیگر از امکانات ارتباطی شبکه‌ها، **کنفرانس ویدئویی** (video conferencing) است. به کمک این تکنولوژی، کارمندانی که هزاران کیلومتر از هم فاصله دارند، می‌توانند یکدیگر را ببینند، صدای هم را بشنوند، و یا حتی مطالب خود را روی یک تخته سیاه مجازی بنویسند. کنفرانس ویدئویی جانشین بسیار مناسبی برای کنفرانسهای واقعی (که متضمن تحمل هزینه‌های سفر است) می‌باشد. گاهی گفته می‌شود که صنعت ارتباطات و حمل و نقل با یکدیگر مسابقهٔ مرگ وزندگی گذاشته‌اند، و هر کدام پیروز شود، دیگری را از میدان بدر خواهد کرد.

اتفاق دیگری که این روزها شتاب بیشتری گرفته، امکان تجارت الکترونیک بین شرکتهای کوچک و بزرگ است. برای مثال، سازندگان کامپیوتر، اتومبیل و هواپیما می‌توانند قطعات مورد نیاز خود را از طریق شبکه‌های کامپیوتری به سازندگان این نوع قطعات سفارش دهند، و سپس آنها را مونتاژ و تبدیل به محصول نهایی کنند. سفارش و خرید قطعات در لحظهٔ نیاز (زمان واقعی) لزوم نگهداری و انبار کردن مقدار زیادی از آنها را منتفی می‌کند.

گرایش تجاری دیگری که حتی اهمیت بیشتری پیدا کرده، فروش محصولات روی اینترنت است. این روزها شرکتهای بسیاری (از قبیل خطوط هوایی، کتابفروشیها، و فروشندگان محصولات فرهنگی) به فروش محصولات خود از طریق اینترنت روی آورده‌اند. این شاخه از تجارت (که به **تجارت الکترونیک** - electronic commerce یا e-commerce - معروف است) در آینده رشد بسیار بیشتری خواهد کرد.



شکل ۱-۲. مدل مشتری-سرویس‌دهنده بر «درخواست و پاسخ» مبتنی است.

۱-۱-۲ کاربردهای خانگی

سال ۱۹۷۷، وقتی از کِن اولسِن (رئیس شرکت Digital Equipment Corporation - که پس از IBM بزرگترین شرکت کامپیوتری دنیا محسوب می‌شد) پرسیدند چرا وارد بازار کامپیوترهای شخصی نمی‌شود، وی پاسخ داد: "هیچ دلیلی ندارد که هر کس توی خانه‌اش یک کامپیوتر داشته باشد." تاریخ ثابت کرد که اولسِن اشتباه می‌کرد، و اکنون دیگر شرکت DEC وجود خارجی ندارد. اما چرا مردم برای کارهای خانگی خود کامپیوتر می‌خرند؟ نوشتن نامه، مقاله و حتی کتاب (و تا یادم نرفته، بازی) یکی از مهمترین دلایل آن است؛ اما این وضعیت امروزه در حال تغییر است. شاید مهمترین دلیل خرید کامپیوترهای خانگی در سالهای اخیر اینترنت باشد. کارهای که این قبیل افراد با کامپیوتر خود انجام می‌دهند، عمدتاً عبارتند از:

۱. دسترسی به اطلاعات پراکنده در سراسر دنیا
۲. ارتباطات دو جانبه
۳. سرگرمیهای تعاملی
۴. تجارت الکترونیک

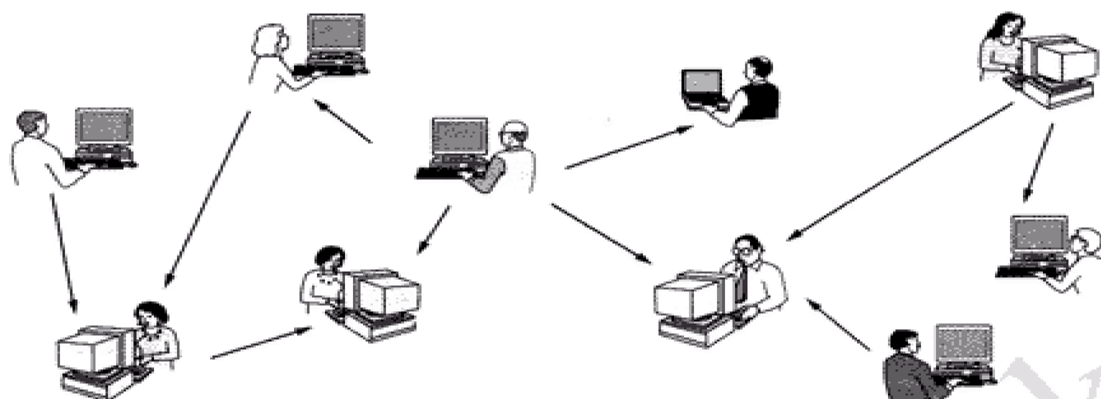
امروزه منبع بسیار عظیمی از اطلاعات در تمامی زمینه‌ها (از قبیل هنر، تجارت، آشپزی، بهداشت، تاریخ، سرگرمی، علم، ورزش و تفریحات سالم - و البته گاهی ناسالم!) روی اینترنت وجود دارد، که می‌توان به آنها دسترسی پیدا کرد. روزنامه‌های بسیاری روی اینترنت منتشر می‌شوند، که می‌توان اخبار را بدلخواه و بصورت گزینشی از آنها بدست آورد. حتی می‌توانید کاری کنید که مقالهٔ دلخواه شما وقتی خواب هستید، از اینترنت بار شده و سپس چاپ شود، تا موقع صبحانه بتوانید با خیال راحت آنرا بخوانید. (به این ترتیب روزنامه‌فروشهای بیچاره بیکار خواهند شد، ولی مطبوعات هم هیچ وقت دل خوشی از آنها نداشتند.)

بعد از روزنامه‌ها و مجلات الکترونیکی نوبت کتابخانه‌های دیجیتالی است. بسیاری از سازمانهای علمی معتبر مانند ACM (www.acm.org) و IEEE (www.computer.org) مدتهاست که انتشارات و کنفرانسهای متعددی روی اینترنت برگزار می‌کنند؛ و این روند بسرعت در حال گسترش است. بنظر می‌رسد که رواج کتابخوانی اینترنتی فقط به قیمت، اندازه و وزن کامپیوترهای کتابی بستگی دارد. (شاید هنوز عده‌ای به این آینده شک داشته باشند؛ اما بهتر است آنها بلایی را که دستگاه چاپ گوتنبرگ سر کتابهای زیبای خطی آورد، بیاد بیاورند.)

تمام کاربردهایی که در بالا نام بردیم، متضمن ارتباط فرد با یک منبع اطلاعات روی اینترنت بود. اما روش دیگری نیز برای برقراری ارتباط از طریق اینترنت وجود دارد، و آن ارتباط فرد به فرد است (این پاسخ تکنولوژی قرن بیست و یکم است به تلفن قرن نوزدهمی). امروزه میلیونها نفر در سراسر دنیا بطور روزمره از ایمیل استفاده می‌کنند؛ صوت و تصویر هم مدتهاست به جزیی جدایی‌ناپذیر از آن تبدیل شده است (و باید بزودی منتظر بوهای اینترنتی هم باشیم!).

این روزها همهٔ نوجوانانی که سری میان سرها در می‌آورند، به برنامه‌های **پیام‌رسان فوری** (instant messaging) معتاد هستند. این برنامه‌ها (که از برنامهٔ talk در سیستمهای یونیکس مشتق شده‌اند) به افراد امکان می‌دهند پیامهای متنی خود را بلافاصله (و بدون تأخیر زمانی) با هم مبادله کنند. نسخه‌هایی از این برنامه‌ها که به چندین نفر اجازه می‌دهند تا در آن واحد با هم گفتگو کنند، به **اتاق گفتگو** (chat room) معروفند.

گروههای خبری (newgroup) از امکانات قدیمی و پرطرفدار اینترنت است، که امکان بحث دربارهٔ موضوعات بسیار متنوعی را به شما می‌دهد. در این سیستم پیامی که می‌فرستید، به تمام آنهایی که عضو گروه خبری هستند خواهد رسید (خوششان بیاید، یا نیاید). مبادلات گروه خبری (بر خلاف اتاق گفتگو) بصورت بلافاصله و در زمان واقعی نیست، و پیامها در نقطه‌ای ذخیره می‌شوند، تا کاربر بتواند هر زمان که خواست آنها را بخواند.



شکل ۱-۳. در یک سیستم همتا-به-همتا مشتری یا سرویس‌دهنده ثابتی وجود ندارد.

نوع دیگری از امکانات ارتباطی موجود در اینترنت، ارتباط **همتا-به-همتا** (peer-to-peer) است. این مدل تفاوت اساسی با مدل مشتری-سرویس‌دهنده دارد (به Parameswaran et al., 2001 نگاه کنید). در این مدل ارتباط افراد در یک گروه غیرثابت و ناپایدار صورت می‌گیرد (شکل ۱-۳ را ببینید). در واقع، هر فرد می‌تواند مستقیماً با هر فرد (یا افراد) دیگر تماس برقرار کند، و چیز ثابتی بعنوان سرویس‌دهنده یا مشتری وجود ندارد.

بزرگترین نمونهٔ ارتباط همتا-به-همتا در حوالی سال ۲۰۰۰ با سرویسی بنام Napster شکل گرفت؛ این سرویس در اوج خود امکان ارتباط بیش از ۵۰ میلیون نفر را فراهم می‌آورد، که بصورت غیر قانونی به رد و بدل کردن موزیک مبادرت می‌کردند (این بزرگترین نقض حق‌التألیف در تاریخ موسیقی بود؛ به Lam and Tan, 2001 و Macedonia, 2000 نگاه کنید). ایدهٔ کار نسبتاً ساده بود: هر نفر می‌توانست آهنگهایی را که در هارد دیسک خود داشت، در پایگاه دادهٔ مرکزی Napster ثبت کند؛ افرادی هم که بدنبال آهنگ خاصی بودند، این پایگاه داده را جستجو کرده، و بعد مستقیماً به سراغ آن می‌رفتند. Napster ادعا می‌کرد که هیچ حق‌التألیفی را نقض نمی‌کند، چون اساساً آهنگها در کامپیوترهای آن ذخیره نمی‌شوند؛ اما دادگاه با این نظر موافق نبود، و حکم به تعطیلی آن داد.

سیستمهای همتا-به-همتا جدید با هوشتر شده‌اند، چون پایگاه دادهٔ مرکزی را حذف کرده‌اند و بجای آن این اطلاعات در کامپیوتر تک تک افراد ذخیره می‌شود، و آنها لیستی از افراد مجاور خود را هم در اختیار دارند. در این روش جستجو کمی بیشتر طول می‌کشد (که بار آن هم بر دوش کامپیوترهاست)، ولی در نهایت به همان اندازه مؤثر است.

همهٔ برنامه‌های همتا-به-همتا هم غیرقانونی نیستند. برای مثال، برنامه‌هایی هستند که اجازه می‌دهند تا آهنگها و فیلمهای مجاز و حتی عکسهای خانوادگی خود را روی اینترنت به اشتراک بگذارید، و یا بازیهای دسته‌جمعی انجام دهید. در حقیقت، پرطرفدارترین کاربرد اینترنت، یعنی ایمیل، ذاتاً یک سیستم همتا-به-همتا است، و بنظر می‌رسد در آینده این سیستمها حتی گسترده‌تر شوند.

جرایم الکترونیکی به دزدی آثار و نقض حق‌التألیف محدود نمی‌شود، و این روزها قمارخانه‌ها هم پایشان به اینترنت باز شده است. کامپیوترها قادرند هر کاری انجام دهند، پس چرا قمار نکنند؟ البته قمار در بسیاری از کشورها غیرقانونی است، اما مشکل اینجاست که در چند جا هم قانونیست (مانند انگلستان)، و صاحبان کازینوها به امکانات بالقوهٔ اینترنت برای قمار واقف شده‌اند. اما اگر کازینو و قمارباز در دو کشور متفاوت (که قوانین متفاوتی هم در زمینهٔ قمار دارند) باشند، چطور؟ سؤال خوبیست!

ارتباطات اینترنتی در زمینهٔ تماسهای تلفنی، ویدئویی و رادیو نیز تحولات وسیعی ایجاد کرده‌اند. **آموزش از**

راه دور (telelearning) نیز یکی دیگر از امکاناتیست که اینترنت عرضه کرده است. (تصورش را بکنید که ساعت ۸ صبح سر کلاس درس حاضر باشید، بدون اینکه لازم باشد قبل از آن از رختخواب بیرون بیایید!) بنظر می‌رسد در دراز مدت اینترنت بزرگترین نقش را در بهبود ارتباطات انسانی بازی کند.

سومین دسته از کاربردهای خانگی شبکه‌های کامپیوتری، صنعت سرگرمی و تفریحات (با رشدی سرسام‌آور) است. داغترین بحث در این زمینه **پخش فیلم برحسب تقاضا** (video on demand) است. شاید تا ده سال دیگر براحتی بتوانید فیلم دلخواه خود را انتخاب کرده، و همان لحظه روی صفحهٔ تلویزیون تماشا کنید. فیلمهای جدید **تعاملی** (interactive) خواهند بود، بدین معنا که بیننده می‌توانند مسیر سناریو را بدلخواه خود تغییر دهد. تلویزیون زنده (شرکت مستقیم و بلافاصله در مسابقات و شوهای تلویزیونی) نیز یکی دیگر از امکانات آینده است.

بازیهای تعاملی یکی دیگر از امکانات شبکه است که شاید آیندهٔ آن حتی از پخش فیلم برحسب تقاضا نیز داغتر باشد. حتی همین حالا هم گروههای بزرگی از جوانان ماجراجو شب و روز مشغول بازی موش و گربه و جنگهای هوایی، زمینی و دریایی در زوایای تاریک و دورافتادهٔ این دنیای مجازی (اینترنت) هستند. اگر آینده بتواند امکانات پخش سه بُعدی و کیفیت بالا را عرضه کند، دیگر این دنیای مجازی هیچ چیز از دنیای واقعی کم نخواهد داشت.

چهارمین دسته از کاربردهای شبکه، شاید وسیعترین آنها باشد: **خرید از خانه** (home shopping). امروزه میلیونها نفر در سراسر جهان هر روز مایحتاج خود را بطور مستقیم از اینترنت تهیه می‌کنند، و هرگز پا از خانه بیرون نمی‌گذارند (حداقل برای خرید). هزاران شرکت بزرگ و کوچک کاتالوگ محصولات خود را بصورتی جذاب روی اینترنت گذاشته‌اند، و برای خرید هر یک از آنها کافیست روی جنس موردنظر یک کلیک کنید. کالایی را خریده‌اید، ولی نمی‌دانید چطور کار می‌کند؟ نگران نباشید، باز هم اینترنت به شما کمک می‌کند، و هر اطلاعات و راهنمایی که بخواهید در اختیارتان قرار می‌دهد.

می‌خواهید صورتحسابهای خود را پرداخت کنید؟ از آخرین وضعیت حسابهای بانکی خود مطلع شوید؟ و یا سرمایه‌گذاری جدیدی بکنید؟ باز هم اینترنت در خدمت شماست. امروزه میلیونها نفر در سراسر جهان کارهای مالی و بانکی خود را بصورت الکترونیکی انجام می‌دهند، و با تقویت مسائل امنیتی شبکه این روند حتی گسترش بیشتری نیز خواهد یافت.

یکی از زمینه‌هایی که شاید هیچکس تصور اینترنتی شدن آنرا نمی‌کرد، سمساری بود. حراج اینترنتی اشیاء دست دوم اینک به یکی از تجارتهای بزرگ تبدیل شده است. بر خلاف تجارت الکترونیک معمولی که از مدل مشتری-سرویس‌دهنده استفاده می‌کند، حراج اینترنتی در واقع یک سیستم همتا-به-همتا یا **خریدار-به-خریدار** (consumer-to-consumer) است. امروزه با اصطلاحات زیادی از این دست برخورد می‌کنید، که در آنها بجای "to" از "2" استفاده می‌شود (چون تلفظ آنها یکسان است). در شکل ۱-۴ تعدادی از رایجترین این اصطلاحات را ملاحظه می‌کنید.

شکی نیست که کاربردهای شبکه و اینترنت در آینده بسرعت افزایش خواهد یافت، و در زمینه‌هایی رسوخ خواهد کرد که امروز حتی به تصور کسی نمی‌آید. چه کسی در سال ۱۹۹۰ می‌توانست تصور کند که بخش بزرگی از درآمد شرکتهای تلفن از محل پیامهای کوتاهی باشد که دانش‌آموزان دبیرستانی بطور خستگی‌ناپذیر و در حالیکه سوار اتوبوس مدرسه هستند، با تلفن همراه خود برای دوستانشان می‌فرستند؟ این شرکتها بخوبی می‌دانند که **سرویس پیام کوتاه** (Short Message Service - SMS) بسیار سودآور است.

شبکه‌های کامپیوتری برای افرادی که دور از شهرها زندگی می‌کنند، نیز مفید است.اینان می‌توانند راحت در

اصطلاح	نام کامل	مثال
B2C	فروشنده-به-خریدار	خرید کتاب روی اینترنت
B2B	فروشنده-به-فروشنده	خرید قطعات یدکی توسط تولیدکننده
G2C	دولت-به-خریدار	توزیع فرمهای مالیاتی از طریق اینترنت
C2C	خریدار-به-خریدار	حراج اشیاء دست دوم
P2P	همتا-به-همتا	اشتراک فایل

شکل ۱-۴. برخی از انواع تجارت الکترونیک.

روستاهای خود زندگی کنند، و در عین حال به تمام امکانات شهرهای بزرگ هم دسترسی داشته باشند. دانشگاههای آینده به احتمال زیاد حالت ملی و محلی خود را از دست داده، و بصورت بین‌المللی درخواهند آمد. **درمان از راه دور** (telemedicine) امروزه به کنترل بیماران محدود می‌شود، ولی چه کسی می‌توانند امکانات بالقوهٔ آنرا پیش‌بینی کند. (یا مثلاً، چقدر خوب می‌شد اگر می‌توانستیم یک دوربین دیجیتالی در یخچال خود نصب کنیم، تا هر وقت شیر تمام شد بتوانیم سر راه خانه شیر بخریم!)

۱-۱-۳ کاربران سیّار

کامپیوترهای سیّار، مانند کامپیوترهای کتابی و **دستیاران دیجیتالی** (PDA)، یکی از سریعترین رشدها را در صنعت کامپیوتر تجربه می‌کنند. اغلب دارندگان این وسایل میل دارند حتی وقتی از خانه دور و یا در سفر هستند، با کامپیوتر خانگی یا دفتری خود ارتباط داشته باشند. در این قبیل موارد دیگر شبکه‌های کابلی محلی از إعراب ندارد، و باید به فکر **شبکه‌های بیسیم** (wireless network) باشیم. در این قسمت نگاهی به کاربردهای شبکه‌های بیسیم خواهیم داشت.

جالبترین کاربرد شبکه‌های بیسیم در ایجاد دفاتر سیّار است. اغلب افراد میل دارند در سفر همان کارهایی را انجام دهند که در دفتر کار خود انجام می‌دهند (ایمیل و فکس بفرستند، تلفن راه دور بزنند، فایلهای خود را باز کنند، و یا در وب گشت بزنند)، و اصلاً هم کاری به این ندارند که کجا هستند! برای مثال، امروزه در اغلب کنفرانسهای کامپیوتری گردانندگان کنفرانس یک شبکهٔ بیسیم در محوطهٔ کنفرانس راه می‌اندازند، و هر کسی می‌تواند با استفاده از یک مودم بیسیم به اینترنت دسترسی پیدا کند. بسیاری از دانشگاهها هم در محوطهٔ خوابگاهی خود شبکه‌های بیسیم دارند، و به دانشجویان امکان می‌دهند تا زیر درختان محوطه نشسته و ایمیل‌های خود را چک کنند، و یا در کتابخانهٔ دانشگاه دنبال مقاله بگردند.

شبکه‌های بیسیم در امور حمل و نقل (کشتیها، کامیونها و تاکسیها) تحولی بزرگ ایجاد کرده‌اند. برای مثال، در بسیاری از شهرهای بزرگ رانندگان تاکسی مستقل بوده و عضو هیچ شرکت یا اتحادیه‌ای نیستند. وقتی کسی به تاکسی نیاز دارد، به یک سرویس مرکزی تلفن می‌کند، و مشخصات وی (از قبیل مبدأ و مقصد) توسط این سرویس به تمام تاکسیها ارسال می‌شود. اولین تاکسیی که مایل به انجام این سرویس باشد، با فشار یک دکمه اعلام آمادگی کرده و به سراغ مسافر می‌رود.

شبکه‌های بیسیم از نظر نظامی نیز اهمیت فوق‌العاده‌ای دارند. هیچ ارتشی نمی‌تواند در جنگهای بزرگ به شبکه‌های عمومی تکیه کند، و بهتر است شبکه‌ای خاص خود بر پا کند؛ و چه چیزی بهتر از یک شبکهٔ بیسیم.

با وجود شباهتهای بسیار بین شبکه‌های بیسیم و کامپیوترهای سیّار، آنها یکی نیستند (شکل ۱-۵ را ببینید). به تفاوت **شبکه‌های بیسیم ثابت** (fixed wireless) و **شبکه‌های بیسیم سیّار** (mobile wireless) توجه کنید. در بسیاری از دفاتر، کامپیوترهای کتابی سیّار بصورت ثابت به شبکهٔ محلی متصل شده‌اند؛ از طرف دیگر کامپیوتری که با استفاده از مودم به شبکه وصل می‌شود، سیّار است - ولی مسلماً به آن بیسیم نمی‌توان گفت.

بیسیم	سیّار	کاربردها
No	No	کامپیوترهای رومیزی در دفتر کار
No	Yes	کامپیوتری که با مودم به شبکه وصل شده
Yes	No	شبکهٔ بیسیم مستقر در یک ساختمان
Yes	Yes	دفاتر سیّار؛ PDA ها

شکل ۱-۵. ترکیب شبکه‌های بیسیم و کامپیوترهای سیّار.

از طرف دیگر، هر شبکهٔ بیسیمی الزاماً سیّار نیست. ساختمانهای بسیاری وجود دارند، که بدلیل مشکلات کابل‌کشی از شبکه‌های بیسیم استفاده می‌کنند. امروزه نصب شبکه‌های بیسیم بسیار ساده شده است، و دردسرهای کابل‌کشی را هم ندارد.

البته ترکیب بیسیم با کامپیوترهای سیّار نیز عملیست، و امروزه کاربردهای مهمی دارد. کسانی که در انبارهای بزرگ یا فرودگاههای شلوغ کار می‌کنند، تجربهٔ استفاده از این سیستمهای ترکیبی را دارند. این کامپیوترها با اطلاعات ورودی کمی که می‌گیرند، و با اتصال بیسیم به پایگاه دادهٔ مرکزی، کار خود را بسرعت و دقت انجام می‌دهند. با رشد تکنولوژی بیسیم، مسلماً کاربردهای آن نیز گسترش خواهد یافت. اجازه دهید نگاهی به این احتمالات بیندازیم.

پارکومترهای بیسیم کار دولت و افراد را راحتتر خواهند کرد. این پارکومترها می‌توانند کارت اعتباری نیز قبول کرده، و بسرعت اعتبار آنرا چک کنند؛ و وقتی مدت پارک تمام شد، اگر هنوز اتومبیل شما آنجا بود، نزدیکترین پلیس را خبر می‌کنند تا آنرا جریمه کند! تخمین زده شده که فقط پلیس ایالات متحده می‌تواند از این طریق ۱۰ میلیارد دلار بر درآمد خود بیفزاید (Harte et al., 2000). این روش اثرات مثبت زیست‌محیطی نیز دارد، چون رانندگان خودروها مطمئنند که راهی برای فرار از دست پلیس ندارند، و به ناچار به وسایل نقلیهٔ عمومی روی می‌آورند.

امروزه ماشینهای خودکار فروش غذا، نوشابه و چیزهای دیگر در همه جا یافت می‌شوند. اما غذا و نوشابه که از آسمان وارد این ماشینها نمی‌شود؛ یک مأمور سوار بر کامیون هر از چند گاهی به این ماشینها سرکشی می‌کند، تا در صورت نیاز آنها را پُر کند. اگر این ماشینها هر روز موجودی خود را از طریق شبکهٔ بیسیم به مرکز اطلاع دهند، مأمور ما می‌داند سراغ کدام ماشینها باید برود، و چه مقدار کالا احتیاج دارد (و حتی می‌تواند مسیر حرکت خود را به بهترین نحو برنامه‌ریزی کند). البته این اطلاعات را از طریق خطوط تلفن هم می‌توان به مرکز منتقل کرد؛ اما کشیدن یک خط تلفن برای هر ماشین (آن هم برای یک تماس در روز) اصلاً مقرون بصرفه نیست.

زمینهٔ دیگری که شبکه‌های بیسیم می‌توانند باعث صرفه‌جویی شوند، قرائت کنتورهای مختلف خانگی است. اگر هر کنتور آب، برق و گاز اطلاعات خود را از طریق شبکهٔ بیسیم به شرکت مربوطه منتقل کند، دیگر نیازی به مراجعهٔ کنتورخوان‌ها به درب منازل نیست. به همین ترتیب، اگر آشکارسازهای دود و حرارت آلارم خود را (بجای راه انداختن آژیر و سر و صدا) مستقیماً به مراکز آتش‌نشانی بفرستند، بسیار مؤثرتر خواهد بود. با کاهش قیمت دستگاههای رادیویی (که مبنای شبکه‌های بیسیم هستند)، وسایل اندازه‌گیری و گزارش‌دهی بیشتری به استفاده از آنها روی خواهند آورد.

زمینهٔ دیگری از کاربردهای شبکه‌های بیسیم (که از مدتها قبل نیز انتظار آن می‌رفت)، ادغام تلفنهای همراه و PDA ها با کامپیوترهای بیسیم است. اولین PDA های بیسیم قادر بودند صفحات سادهٔ وب را روی صفحات کوچک خود نمایش دهند. این سیستم که WAP 1.0 (Wireless Application Protocol) نام داشت، بدلیل خوانا نبودن صفحات، پهنای باند کم، و سرویس ضعیف با شکست مواجه شد. ولی با سرویسها و وسایل جدید WAP 2.0 اوضاع مسلماً بهتر خواهد شد.

ترکیب این تکنولوژیها می‌تواند به سرویس جدیدی منجر شود که می‌توان آنرا **تجارت سیّار** (mobile commerce) نامید (Senn, 2000). این پدیده در واقع عامل ترکیب‌کنندهٔ تکنولوژی PDA های بیسیم با تجارت الکترونیک است، که این روزها همه از آن سهم می‌خواهند، و این امیدواری وجود دارد که افراد به خرید و انجام کارهای بانکی با آن روی آورند. PDA بیسیم می‌تواند در فروشگاهها و مراکز خرید بعنوان عامل انتقال پول و یا کارت اعتباری عمل کند (بدین ترتیب که این هزینه‌ها بعداً با صورتحساب تلفن پرداخت شود). نکتهٔ مثبت این روش برای فروشندگان آن است که هزینه‌های کار با شرکتهای اعتباری را به مقدار زیادی پائین می‌آورد. البته عیب بزرگی نیز دارد: خریداران می‌توانند قبل از خرید با PDA بیسیم خود قیمتها را با فروشندگان دیگر مقایسه کنند. اگر شرکتهای تلفن سرویس قرائت بارکُد را هم به این PDA ها اضافه کنند، که دیگر اوضاع خرابتر می‌شود! چون دیگر حتی با مخفی کردن قیمتها هم نمی‌توان جلوی مقایسهٔ قیمتها توسط خریدار را گرفت (کاری که خیلی از فروشندگان به آن امید بسته‌اند)!

از آنجائیکه محل این دستگاهها همیشه برای اپراتور سیستم مشخص است، می‌توان سرویسهای خاصی را در اختیار کاربران آنها گذاشت؛ مثلاً، می‌توان آدرس نزدیکترین کتابفروشی یا رستوران چینی را در اختیار وی گذاشت، و یا آخرین پیش‌بینی وضعیت هوا را به وی اعلام کرد.

امکانات و کاربردهای این سرویس جدید می‌تواند بسیار فراتر از مثالهای سادهٔ فوق باشد؛ و خوبی قضیه اینست که کاربران تلفنهای موبایل عادت دارند برای هر چیزی پول بدهند (درست بر خلاف کاربران اینترنت که همه چیز را مجانی می‌خواهند)! اگر یک سایت اینترنتی برای قبول کارت اعتباری از شما درخواست پول کند، فوراً فریادتان بلند خواهد شد، ولی اگر همین اتفاق روی سرویسهای موبایل بیفتد، بدون هیچ اعتراضی قبول می‌کنید (البته فعلاً).

بگذارید کمی هم به آینده نگاه کنیم: **شبکه‌های شخصی** (Personal Area Network) ، و **کامپیوترهای پوشیدنی** (wearable computer) . به تازگی IBM ساعتی ساخته که سیستم عامل **لینوکس** (Linux) روی آن اجرا می‌شود، و می‌تواند به اینترنت وصل شده و ایمیل رد و بدل کند (Narayanaswami et al., 2002). در آینده دیگر چیزی بعنوان کارت ویزیت بین افراد رد و بدل نخواهد شد، و آنها می‌توانند با یک تماس ساعت مچی تمام اطلاعات طرف مقابل را دریافت کنند. به احتمال زیاد کامپیوترهای پوشیدنی (که اطلاعات زیستی فرد را در خود دارند) جای کارتهای مغناطیسی را برای ورود به مکانهای حساس خواهند گرفت، و یا می‌توانند در هر لحظه مکان فرد را اعلام کنند. امکانات این سیستمها تقریباً بیشمار است.

ساعتها و رادیوهای هوشمند را همهٔ ما سالهاست از طریق فیلمهای جیمز باند می‌شناسیم، ولی آیا تا بحال گرد و غبار هوشمند به گوشتان خورده است؟ محققان دانشگاه برکلی اخیراً یک کامپیوتر بیسیم ساخته‌اند، که در مکعبی با ابعاد 1 mm جای می‌گیرد (Warneke et al., 2001). با این کامپیوترها می‌توان مسیر حرکت چمدان در فرودگاه‌ها، و یا جانوران (مثلاً، پرندگان مهاجر) را با دقت کنترل کرد.

۱-۱-۴ تبعات اجتماعی

گسترش روزافزون شبکه‌های کامپیوتری باعث ایجاد مسائل اجتماعی، اخلاقی و سیاسی خاص خود شده است که در این قسمت برخی از آنها را بررسی می‌کنیم. یکی از امکانات شبکه‌های کامپیوتری تبادل آزاد و سریع اطلاعات و اخبار است. البته تا وقتی این پیامها در محدوده‌های فنی باقی بماند، مشکل چندانی وجود نخواهد داشت؛ دردسر وقتی شروع می‌شود که صحبت به مسایل حساس (از قبیل سیاست، مذهب یا سکس) کشیده شود.

نظرها و دیدگاههایی که توسط اعضای یک گروه خبری پُست می‌شود، ممکنست برای افراد دیگر بسیار ناهنجار و موهن باشد؛ و وقتی پیامها به متن محدود نشود، کار بدتر هم خواهد شد. امروزه براحتی می‌توان

عکسهای بسیار واضح و باکیفیت عالی (و یا حتی کلیپهای ویدئویی کوچک) را از طریق اینترنت منتشر کرد. برخی از افراد در زندگی به فلسفهٔ «زندگی کن، و بگذار زندگی کنند» معتقدند، اما عدهٔ زیادی هم هستند که احساس می‌کنند به برخی مطالب (مانند حمله به کشورها یا مذاهب دیگر، صور قبیحه و غیره) نباید اجازهٔ انتشار داد. کشورهای مختلف هم دارای قوانین متفاوتی در این زمینه‌ها هستند؛ و جدال از همین جا شروع می‌شود.

در این میان بسیاری از افراد اپراتورهای شبکه را (مانند روزنامه‌ها و مجلات) مسئول محتویات شبکه می‌دانند، اما واقعیت اینست که یک شبکه بیشتر شبیه ادارهٔ تلفن و پُست است تا روزنامه یا مجله (و نمی‌توان آنرا مسئول چیزهایی که از این طریق مبادله می‌شود، دانست). از طرف دیگر، اگر اپراتور شبکه اجازه داشته باشد مطالب را سانسور کند، به احتمال زیاد (برای فرار از متهم شدن) روی کوچکترین چیزها هم انگشت خواهد گذاشت، و بدین ترتیب حقوق افراد در زمینهٔ آزادی بیان از بین خواهد رفت. بحث موافق و مخالف همچنان ادامه دارد (و براحتی می‌توان حدس زد که به این زودی‌ها هم به نتیجه نخواهد رسید).

بحث جالب دیگر حقوق و رابطهٔ کارگر و کارفرما است. بسیاری از افراد در محل کار خود ایمیل می‌فرستند، و یا ایمیل‌های رسیده را می‌خوانند. برخی از کارفرمایان ادعا می‌کنند که آنها حق دارند ایمیل‌های کارمندان خود را بخوانند و یا آنها را سانسور کنند؛ و صد البته کارمندان با این حرفها موافق نیستند!

حتی اگر بپذیریم کارفرمایان در ادعای خود محق هستند، آیا می‌توان این رابطه را به دانشجو و دانشگاه (و یا دانش‌آموز و مدرسه) تعمیم داد؟ در سال ۱۹۹۴ دانشگاه کارنگی-ملون تصمیم گرفت برخی از پیامهای رسیده را که در آنها به موضوعات سکسی پرداخته شده بود، سانسور کند (با این استدلال که این مطالب برای افراد زیر ۱۸ سال مناسب نیست). سالها طول کشید تا پس لرزه‌های این اقدام فروکش کند.

و از همه مهمتر رابطهٔ دولت با شهروندان است. ادارهٔ آگاهی فدرال ایالات متحده (FBI) سالهاست سیستمی را در مراکز **ارائهٔ سرویس اینترنت** (ISP) نصب کرده، که به آن اجازه می‌دهد تا ایمیلهای ورودی و خروجی را تجسس کند (Blaze and Bellovin, 2000 ؛ Sobel, 2001 ؛ و Zacks, 2001). نام این سیستم Carnivore (گل گوشتخوار) بود، که بدلیل حساسیتهای ایجاد شده در جامعه به نام کم ضررتر DCS1000 تغییر داده شد (با این حال کار آن همچنان جاسوسی در ایمیلهای مردم بود). طبق اصلاحیهٔ چهارم قانون اساسی ایالت متحدهٔ امریکا، دولت بدون مجوز قانونی حق تجسس در احوال شخصی افراد را ندارد. اینکه این قانون نوشته شده در قرن هیجدهم هنوز در قرن بیست و یک اعتبار دارد یا خیر، را آینده روشن خواهد کرد.

فضولی در کار مردم به دولت محدود نمی‌شود؛ بخش خصوصی هم از این گناه در امان نیست. برای مثال، مرورگرهای وب از فایلهای کوچکی بنام **کوکی** (cookie) استفاده می‌کنند که اطلاعات شخصی افراد را در اختیار شرکتها می‌گذارد، و حتی می‌تواند منجر به افشای شمارهٔ کارتهای اعتباری و اطلاعات محرمانهٔ دیگر روی اینترنت شود (Berghel, 2001).

در شبکه‌های کامپیوتری می‌توان پیامهای بدون نام و نشانی فرستاد، که در جای خود می‌تواند مفید باشد. مثلاً، کارمندان، دانشجویان و یا مردم عادی می‌توانند بدین طریق اعمال خلاف رؤسای شرکتها، استادان و سیاستمداران را به اطلاع عموم برسانند بدون آنکه از اقدامات تلافی‌جویانهٔ آنها ترسی به دل راه دهند. (البته در بسیاری از کشورها از این قبیل اطلاعات بدون منبع نمی‌توان در دادگاه بعنوان مدرک جرم استفاده کرد.)

وضعیت فعلی شبکه‌های کامپیوتری شبیه موقعیت کتابهای چاپی در ابتدای اختراع این صنعت بود: افراد عادی وسیله‌ای بدست آورده بودند تا با آن صدای خود را به گوش دیگران برسانند. اما این آزادی با خود تبعات اجتماعی، سیاسی و اخلاقی خاصی بدنبال داشت، که همچنان لاینحل باقی مانده است.

زندگی همیشه بدین منوال است: هر سکه‌ای دو رو دارد. اینترنت هم از این قاعده مستثنی نیست. امکان

دسترسی سریع و آسان به اطلاعات ارمغان اینترنت است، ولی خروارها اطلاعات منفی، غلط و گمراه‌کننده وجه دیگر آن است. راهنمایی بهداشتی که تازگی در اینترنت خوانده‌اید (و احتمالاً می‌خواهید به آن عمل کنید)، ممکنست از یک برندهٔ جایزهٔ نوبل آمده باشد، یا یک دانش‌آموز بازیگوش دبیرستانی.

شبکه‌های کامپیوتری انواع جدیدی از جرم و رفتارهای ضداجتماعی را نیز با خود آورده‌اند. هر روز که صندوق پُستی خود را باز می‌کنید، دهها و صدها پیام مزخرف و بدردنخور در آن می‌بینید (و حتی کم‌کم به آن عادت کرده‌اید). این نتیجهٔ کار افرادیست که میلیونها آدرس ایمیل را روی یک CD جمع کرده، و (بدون رضایت صاحبان این آدرسها) به این و آن می‌فروشند. و تازه اینها دستهٔ بی‌آزارها هستند؛ این روزها کسی پیدا نمی‌شود که صابون ویروسهایی که از طریق ایمیل منتشر می‌شوند، به تنش نخورده باشد.

دزدی هویت یکی دیگر از خطرات سرقت اطلاعات از طریق اینترنت است. (دربارهٔ سرقتهای ادبی و نقض گسترده و وسیع قانون حق‌التألیف در اینترنت قبلاً هم صحبت کردیم.)

بسیاری از این مشکلات حاصل ضعف (و یا عدم رعایت) مسائل امنیتی در اینترنت است. اگر تمام پیامهای ایمیل بصورت رمز در آیند، سرقت اطلاعات بسیار مشکلتر خواهد شد (این تکنولوژی توسعهٔ زیادی یافته، که در فصل ۸ مفصلاً به آن خواهیم پرداخت). مشکل اینجاست که بالا بردن سطح ایمنی مترادف است با بالا رفتن هزینه، و این چیزی نیست که به آسانی پذیرفته شود. تعداد زیادی از این مسایل نیز به مشکلات و باگهای موجود در نرم‌افزارها مربوط می‌شود، که خود حاصل بزرگتر و پیچیده‌تر شدن آنهاست. اگر روی برنامه‌ها بر حسب بزرگی و پیچیدگی آنها مالیات بسته شود، شاید این مشکل تا حدی حل شود؛ البته خیلی‌ها این راه حل را نمی‌پسندند! پس دادن پول برنامه‌های معیوب نیز می‌تواند راه حل خوبی باشد، فقط مشکل اینجاست که چنین قانونی ظرف یک سال کل صنعت نرم‌افزار را ورشکست خواهد کرد!

۲-۱ سخت‌افزار شبکه

اکنون وقت آنست که توجه خود را از مسایل متفرقه به موضوع اصلی (یعنی همان شبکه‌های کامپیوتری) معطوف کنیم. هیچ طبقه‌بندی پذیرفته شده‌ای که در بر گیرندهٔ تمام انواع شبکه‌های کامپیوتری باشد، وجود ندارد، ولی در این میان می‌توان به دو عامل مهم توجه کرد: تکنولوژی انتقال و اندازهٔ شبکه. اجازه دهید این دو را جداگانه بررسی کنیم. امروزه دو تکنولوژی انتقال بیش از همه گسترش یافته و فراگیر هستند:

۱. **ارتباطات پخشی** (broadcast)
۲. **ارتباطات همتا به همتا** (peer-to-peer)

شبکه‌های پخشی (broadcast network) دارای یک کانال مخابراتی هستند که بین همهٔ کامپیوترهای شبکه به اشتراک گذاشته شده است. هر یک از کامپیوترها می‌توانند پیامهای خود را در **بسته** (packet) های کوچک مخابره کنند، و تمام کامپیوترهای دیگر این پیامها را دریافت خواهند کرد. آدرس کامپیوتری که این بسته در حقیقت برای وی ارسال شده، در بخشی از پیام نوشته می‌شود. هر کامپیوتری به محض دریافت بسته، آدرس گیرنده را چک می‌کند: اگر پیام برای او باشد، آنرا پردازش می‌کند؛ ولی اگر پیام متعلق به دیگری باشد، بسادگی آنرا نادیده می‌گیرد.

بعنوان مقایسه، فرض کنید کسی در انتهای راهرویی که در دو طرف آن پُر از اتاقهای متعدد است، فریاد بزند «آقای واتسون، بیائید. با شما کار دارم.» با اینکه این پیام به گوش همهٔ افراد می‌رسد، فقط آقای واتسون به آن پاسخ می‌دهد و دیگران توجهی به آن نخواهند کرد. یا وقتی در سالن انتظار فرودگاه اعلام می‌شود که «مسافران پرواز ۶۴۴ به خروجی ۱۲ مراجعه کنند»، فقط آنهایی که بلیط این پرواز را دارند، عکس‌العمل نشان می‌دهند.

در شبکه‌های پخشی با تعبیهٔ یک کُد خاص در فیلد آدرس (address field) می‌توان یک پیام را به تمام

کامپیوترها ارسال کرد. چنین پیامی را همهٔ کامپیوترها متعلق به خود تلقی کرده، و آنرا می‌خوانند. به این تکنیک **پخش** (broadcasting) گفته می‌شود. در برخی از سیستمهای پخشی امکان ارسال پیام به دسته‌ای از کامپیوترها نیز وجود دارد، که به آن **پخش گروهی** (multicasting) می‌گویند. بدین منظور، معمولاً از یک بیت خاص در فیلد آدرس استفاده می‌شود، و همهٔ آنهایی که این بیت در آنها وجود دارد عضو گروه محسوب شده و پیام را می‌گیرند.

در **شبکه‌های همتا به همتا** (peer-to-peer network) بین تک تک کامپیوترها مسیر ارتباطی مستقل وجود دارد. البته وقتی یک بسته بخواهد از کامپیوتری به کامپیوتر دیگر برود، احتمالاً سر راه خود از چند ماشین بینابینی نیز عبور خواهد کرد. معمولاً در این قبیل شبکه‌ها مسیرهای متعددی بین دو کامپیوتر خاص می‌توان برقرار کرد، که از نظر طول مسیر با هم تفاوت دارند، و یافتن کوتاهترین مسیر یکی از مسایل مهم در این گونه شبکه‌هاست. بعنوان یک قاعدهٔ کلی (البته با استثناهای متعدد)، شبکه‌های کوچک، متمرکز و محلی از نوع پخشی هستند، و شبکه‌های بزرگ و گسترده از نوع همتا به همتا. به ارتباط همتا به همتا گاهی **پخش تکی**(unicasting) نیز گفته می‌شود.

روش دیگر طبقه‌بندی شبکه‌ها اندازهٔ شبکه است. در شکل ۱-۶ نوعی طبقه‌بندی بر اساس اندازه را مشاهده می‌کنید.

در بالا **شبکه‌های شخصی** (Personal Area Network) را می‌بینید (شبکه‌هایی که متعلق به یک فرد خاص هستند). ارتباط بیسیم بین ماوس، کی بورد، چاپگر، PDA و کامپیوتر از این نوع است. بعد از آن شبکه‌های **محلی** (LAN)، **شهری** (MAN) و **گسترده** (WAN) می‌آیند. در آخر هم شبکهٔ شبکه‌ها (شبکه‌ای که هر نقطه از آن خود یک شبکهٔ کامل است) - و اینترنت معروفترین نمونهٔ آن است - می‌آید. در این طبقه‌بندی فاصلهٔ کامپیوترها اهمیت زیادی دارد، چون تکنولوژی ارتباطی به شدت به آن وابسته است. در این کتاب دربارهٔ تمام این شبکه‌ها صحبت خواهیم کرد. در زیر هر یک از این شبکه‌ها را مختصراً معرفی می‌کنیم.

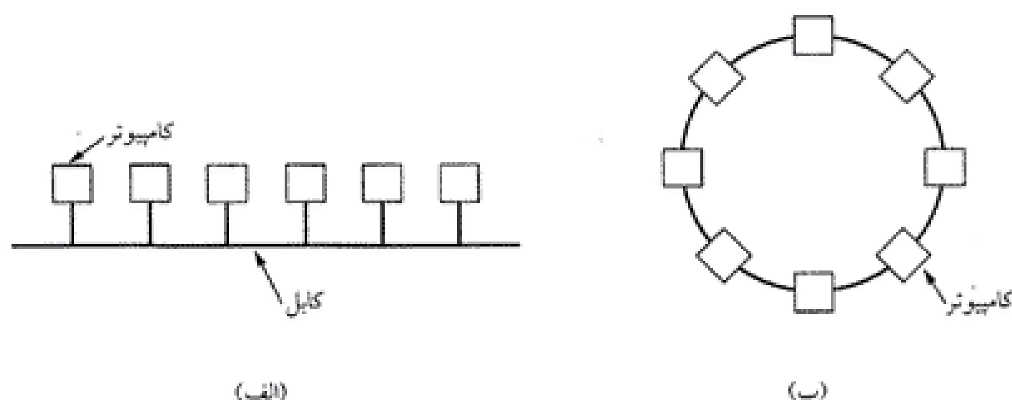
۱-۲-۱ شبکه‌های محلی (Local Area Network)

شبکهٔ محلی، یا LAN، شبکه‌ایست خصوصی واقع در یک ساختمان یا مجتمع، که حداکثر ابعاد آن یکی دو کیلومتر باشد. از این نوع شبکه معمولاً برای متصل کردن کامپیوترهای یک شرکت و به اشتراک گذاشتن منابع (مانند چاپگر) یا مبادلهٔ اطلاعات استفاده می‌شود. یک شبکهٔ LAN سه مشخصهٔ اصلی دارد، که آنرا از سایر انواع شبکه متمایز می‌کند: ۱) اندازه، ۲) تکنولوژی انتقال اطلاعات، و ۳) **توپولوژی** (topology).

اندازهٔ LAN بسیار محدود است، بگونه‌ایکه زمان انتقال سیگنالها در آن (حتی در بدترین شرایط) بسیار کم و از قبل قابل پیش‌بینی است. دانستن این محدودیت‌ها برای طراحی شبکه بسیار مهم و اساسی است، و باعث ساده‌تر شدن مدیریت شبکه نیز می‌شود.

فاصلهٔ پردازنده‌ها	محل نسبی پردازنده‌ها	نمونه
1 m	روی یک میز	شبکهٔ شخصی
10 m	یک اتاق	شبکهٔ محلی
100 m	یک ساختمان	«
1 km	یک مجتمع	«
10 km	یک شهر	شبکهٔ شهری
100 km	یک کشور	شبکهٔ گسترده
1000 km	یک قاره	«
10,000 km	کرهٔ زمین	اینترنت

شکل ۱-۶. طبقه‌بندی شبکه‌ها بر اساس اندازه و فاصلهٔ پردازنده‌ها.



شکل ۱-۷. دو شبکهٔ پخشی. (الف) باس. (ب) حلقوی.

تکنولوژی انتقال اطلاعات در LAN معمولاً به کابل متکیست (و از این نظر بسیار شبیه شبکه‌های تلفن است). سرعت انتقال اطلاعات در LAN بین ۱۰ تا ۱۰۰ میلیون بیت در ثانیه (که با Mbps مشخص می‌شود)، تأخیر انتشار در آن کم (در حد میکرو یا نانوثانیه)، و خطا در آن بسیار اندک است. LAN های جدیدتر به سرعت 10 Gbps نیز دست یافته‌اند. سرعت انتقال در شبکه معمولاً با واحد مگابیت بر ثانیه (1,000,000 bits/sec) یا گیگابیت بر ثانیه (1,000,000,000 bits/sec) اندازه‌گیری می‌شود.

توپولوژی‌های مختلفی برای شبکه‌های محلی پخشی وجود دارد، که در شکل ۱-۷ دو تا از آنها را می‌بینید. در یک **شبکهٔ باس** (bus network - شبکه‌ای با کابل‌کشی خطی) در هر لحظه فقط یکی از کامپیوترها مجاز به استفاده از خط و ارسال اطلاعات است، و تمام ماشینهای دیگر بایستی در این مدت از ارسال هر گونه اطلاعات خودداری کنند. در این قبیل شبکه‌ها بایستی مکانیزمی برای حل اختلاف (در مواقعی که دو کامپیوتر همزمان با هم شروع به ارسال می‌کنند) وجود داشته باشد. این مکانیزم می‌تواند **متمرکز** (centralized) یا **توزیع‌شده** (distributed) باشد. یکی از مکانیزمهای حل اختلاف در شبکه‌های باس پخشی IEEE 802.3 نام دارد (که به **اترنت** - Ethernet - نیز معروف است)، و با کنترل غیرمتمرکز در سرعتهای 10 Mbps تا 10 Gbps کار می‌کند. کامپیوترهای یک شبکهٔ اترنت در هر زمانی می‌توانند اقدام به ارسال کنند، ولی اگر تصادمی بین آنها پیش آمد، هر یک از آنها مدتی (که بصورت تصادفی تعیین می‌شود) صبر کرده و دوباره سعی خواهد کرد.

نوع دیگری از شبکه‌های پخشی، **شبکهٔ حلقوی** (ring network) است. در یک شبکهٔ حلقوی، هر بیت اطلاعات بصورت مستقل (و بدون اینکه بخواهد منتظر سایر بیت‌های بسته‌ای که به آن تعلق دارد، شود) در شبکه منتشر می‌شود. با توجه به سرعت بالای انتشار الکترونها در محیطهای رسانا، هر بیت حتی قبل از انتشار بیت‌های بعدی، می‌تواند بارها محیط شبکه را دور بزند. در این نوع شبکه هم بایستی مکانیزمی برای حل اختلاف بین کامپیوترهای متخاصم وجود داشته باشد. اغلب این مکانیزمها به نوعی نوبت‌بندی متکی هستند. یکی از این مکانیزمها IEEE 802.5 (یا IBM Token Ring) است، که در سرعتهای 4 Mbps و 16 Mbps کار می‌کند. FDDI یکی دیگر از شبکه‌های پخشی حلقوی است.

نوع دیگری از تقسیم‌بندی شبکه‌های پخشی بر حسب نحوهٔ اختصاص کانال است، و به استاتیک و دینامیک تقسیم می‌شود. در اختصاص کانال استاتیک هر کامپیوتر برای مدت زمانی محدود و مشخص کانال را در دست می‌گیرد، و فقط در این بُرش زمانیست که می‌تواند اطلاعات ارسال کند. در این روش پهنای باند کانال بشدت هدر می‌رود، چون بسیار پیش می‌آید که وقتی نوبت به یک کامپیوتر می‌رسد، چیزی برای گفتن ندارد. به همین دلیل، سیستمهای امروزی اغلب پهنای باند را بصورت دینامیک (بر حسب نیاز) تخصیص می‌دهند.

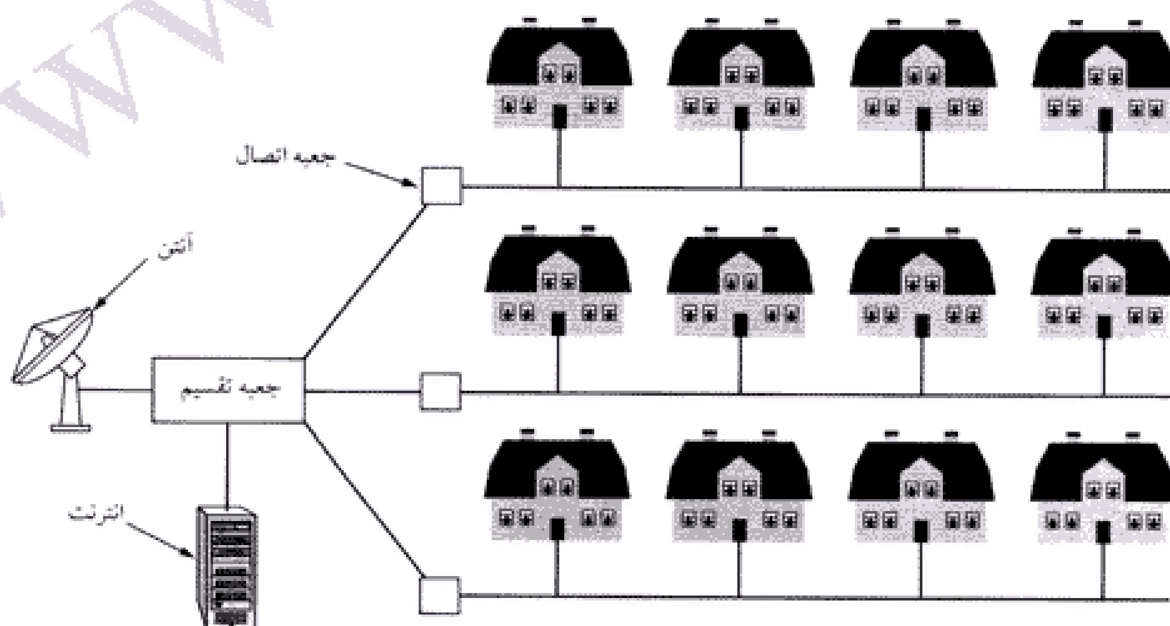
تخصیص دینامیک کانال خود بر دو نوع است: متمرکز و غیرمتمرکز. در نوع متمرکز یک موجودیت مشخص (بنام واحد تصمیم‌گیرنده – Arbitration Unit) وجود دارد، که درخواست‌ها را دریافت کرده، و بر اساس نوعی الگوریتم داخلی نوبت‌ها را تعیین می‌کند. در تخصیص کانال غیرمتمرکز این موجودیت تصمیم‌گیرنده وجود ندارد، و تصمیم‌گیری بر عهدهٔ تک تک کامپیوترهاست. شاید فکر کنید این روش جز هرج و مرج نتیجه‌ای ندارد، ولی چنین نیست (در آینده خواهید دید که الگوریتمهایی وجود دارند که می‌توانند به اوضاع سر و سامان بدهند).

۱-۲-۲ شبکه‌های شهری (Metropolitan Area Network)

شبکهٔ شهری، یا MAN ، شبکه‌ایست که یک شهر را پوشش می‌دهد. شبکه‌های تلویزیون کابلی بهترین نمونهٔ MAN هستند. اولین شبکه‌های تلویزیون کابلی در نقاط کور شهرها راه‌اندازی شدند، بدین ترتیب که یک آنتن مرکزی و بزرگ در محلی که فرستندهٔ اصلی را می‌دید نصب، و از این آنتن کابلهایی به مشترکان محروم از برنامه‌های تلویزیونی کشیده می‌شد.

در ابتدا این سیستمها بطور اختصاصی برای هر محل ساخته می‌شد، ولی بزودی شرکتهای بزرگ بوی پول را از آن استشمام کردند، و با کسب اجازهٔ دولت تمام شهر را زیر پوشش کابلهای خود بردند. این شبکه‌ها برای پخش برنامه هم برنامه‌ریزی خاصی دارند، مثلاً یک شبکه فقط اخبار پخش می‌کند، دیگری فقط برنامه‌های ورزشی دارد، و آن یکی فقط آشپزی. این شبکه‌ها بسیار تخصصی بودند، و تا اواخر دهۀ ۱۹۹۰ فقط برنامه‌های تلویزیونی پخش می‌کردند.

با شروع گرایش عمومی به اینترنت، گردانندگان این شبکه‌ها بزودی دریافتند که با تغییری مختصر در سیستمهای خود می‌توانند از قسمتهای بالااستفادهٔ پهنای باند برای ارائهٔ سرویسهای دوطرفهٔ اینترنت بهره ببرند. از این لحظه بود که شبکه‌های تلویزیون کابلی تبدیل به شبکه‌های شهری (MAN) شدند. در شکل ۱-۸ نمایی تقریبی از یک شبکهٔ شهری را ملاحظه می‌کنید. در این شکل می‌بینید که ابتدا سیگنالهای تلویزیونی و اینترنتی ترکیب شده و به یک مرکز فوق-توزیع (head end) می‌روند، تا از آنجا در خانه‌های مشترکان توزیع شوند. (در فصل ۲ باز هم به این مبحث خواهیم پرداخت.)



شکل ۱-۸ یک شبکهٔ شهری مبتنی بر تلویزیون کابلی.

تلویزیون کابلی تنها مثال زندهٔ MAN نیست. اخیراً تحقیقاتی بر روی **اینترنت بیسیم پُرسرعت** (high-speed wireless Internet) انجام شده، که نتیجهٔ آن نوع دیگری از MAN خواهد بود. با این استاندارد که IEEE 802.16 نام دارد، در فصل ۲ بیشتر آشنا خواهید شد.

۱-۲-۳ شبکه‌های گسترده (Wide Area Network)

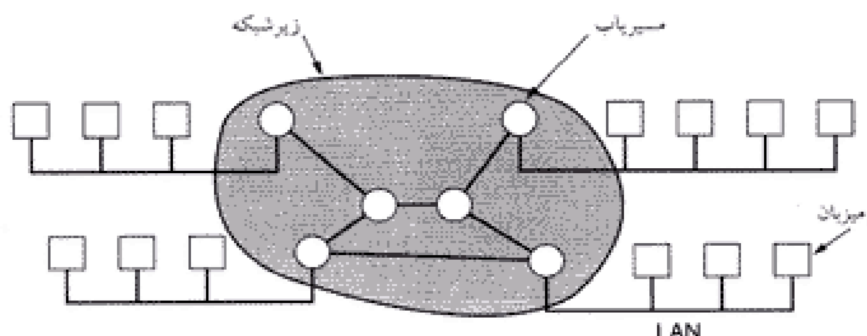
شبکهٔ گسترده، یا WAN ، گسترهٔ جغرافیایی بزرگی (مانند یک کشور یا قاره) دارد. در این نوع شبکه کامپیوترهایی هستند که برنامه‌های کاربردی روی آنها اجرا می‌شود، و معمولاً به آنها **میزبان** (host) می‌گویند. این کامپیوترها توسط **زیرشبکه‌های مخابراتی** (communication subnet) - یا بطور مختصر، **زیرشبکه** - به هم متصل می‌شوند. میزبانها متعلق به افراد هستند، در حالیکه زیرشبکه اغلب به شرکتهای مخابرات تعلق دارد. وظیفهٔ زیرشبکه انتقال پیام از یک میزبان به میزبان دیگر است. جدا کردن این دو بخش (میزبانها و زیرشبکه) طراحی شبکه‌های WAN را تا حد زیادی ساده می‌کند.

در اغلب شبکه‌های گسترده، زیرشبکه از دو بخش مجزا تشکیل می‌شود: **خطوط انتقال** (transmission lines) و **تجهیزات سوئیچینگ** (switching elements) . خطوط انتقال وظیفهٔ رد و بدل کردن اطلاعات را بر عهده دارند، و می‌توان برای ایجاد آنها از سیم مسی، فیبر نوری یا حتی امواج رادیویی استفاده کرد. تجهیزات سوئیچینگ کامپیوترهای خاصی هستند که ارتباط بین خطوط انتقال را برقرار می‌کنند. وقتی داده‌ها از یک خط وارد می‌شود، این کامپیوتر باید مسیر خروجی آنرا مشخص کند. این کامپیوترهای سوئیچینگ به نامهای مختلفی خوانده می‌شوند، که می‌توان از معروفترین آنها به **مسیریاب** (router) اشاره کرد.

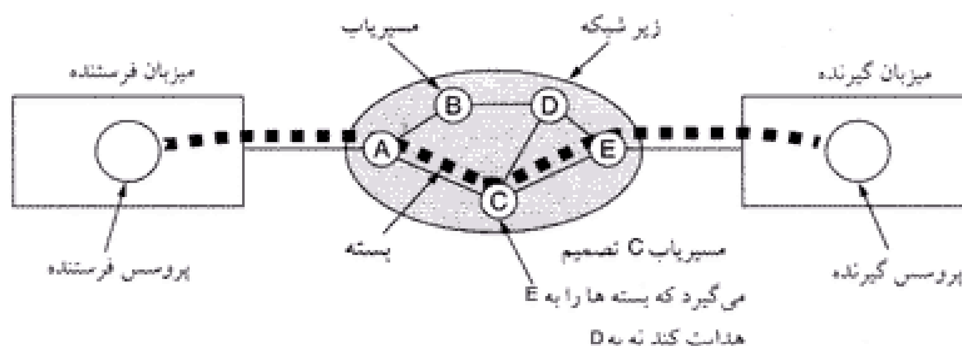
در این مدل (شکل ۱-۹) معمولاً هر کامپیوتر میزبان در یک شبکهٔ محلی قرار دارد که از طریق یک مسیریاب به قسمتهای دیگر متصل می‌شود (البته در مواردی میزبان می‌تواند مستقیماً نیز به مسیریاب وصل باشد). به مجموعهٔ خطوط مخابراتی و مسیریاب‌ها (منهای کامپیوترهای میزبان) زیرشبکه گفته می‌شود.

معنای اولیهٔ زیرشبکه همان است که در بالا گفته شد، یعنی مجموعهٔ خطوط مخابراتی و مسیریاب‌ها که وظیفهٔ آنها انتقال اطلاعات از یک میزبان به میزبان دیگر است. اما سالها بعد از این اصطلاح در ارتباط با آدرس‌دهی شبکه‌ها نیز استفاده شد (فصل ۵ را ببینید). متأسفانه هنوز اصطلاح مناسبی برای کاربرد اولیهٔ آن پیدا نشده است، و ما هم (با کمی شک و تردید) آنرا در هر دو مورد بکار خواهیم برد، که با توجه به موضوع بحث می‌توان معنای موردنظر را استنباط کرد.

در بسیاری از WAN ها تعداد زیادی خطوط انتقال وجود دارد، که هر کدام یک جفت مسیریاب را به هم وصل می‌کنند. اگر دو مسیریاب که اتصال فیزیکی مستقیم ندارند، بخواهند با یکدیگر ارتباط برقرار کنند، باید این کار را بصورت غیرمستقیم (از طریق مسیریاب‌های دیگر) انجام دهند. وقتی یک بستهٔ داده در مسیر خود (از مسیریاب



شکل ۱-۹. ارتباط بین کامپیوترهای میزبان و LAN ها در یک زیرشبکه.



شکل ۱-۱۰. استریم (جریان) بسته‌ها از مبدأ به مقصد.

مبدأ به مسیریاب مقصد) از چند مسیریاب بینابینی عبور می‌کند، ابتدا بصورت کامل دریافت و ذخیره شده، و پس از آزاد شدن خط خروجی به سمت مقصد فرستاده می‌شود. زیرشبکه‌هایی که بر اساس این قاعده عمل می‌کنند، به زیرشبکهٔ **ذخیره-ارسال** (store-and-forward) یا **سوئیچ بسته** (packet-switched) معروفند. تقریباً تمامی شبکه‌های WAN (بجز شبکه‌های ماهواره‌ای) از این نوع هستند. اگر اندازهٔ بسته‌ها کوچک و یکسان باشد، به آنها **سلول** (cell) نیز گفته می‌شود.

به دلیل اهمیت مفهوم زیرشبکهٔ سوئیچ بسته، لازمست کمی بیشتر دربارهٔ آن توضیح دهیم. وقتی پروسسی در یک میزبان می‌خواهد پیامی به میزبان دیگر بفرستد، ابتدا آنرا به بسته‌های کوچکتر (که پشت سر هم شماره‌گذاری می‌شوند) تقسیم می‌کند. این بسته‌ها بصورت مستقل به طرف مقابل ارسال می‌شوند، و بعد از رسیدن تمامی آنها به مقصد، در آنجا دوباره به یکدیگر مونتاژ شده و پیام اصلی را می‌سازند (شکل ۱-۱۰ را ببینید).

در این شکل تمام بسته‌ها از طریق مسیر ACE به مقصد رسیده‌اند (در حالیکه مسیرهای ABDE و ACDE نیز وجود داشت). در برخی از شبکه‌ها این یک الزام است، یعنی تمام بسته‌های یک پیام باید از یک مسیر عبور کنند، در حالیکه در شبکه‌های دیگر این بسته‌ها می‌توانند از مسیرهای مختلف عبور کنند. البته، اگر مسیری بهترین مسیر ممکن باشد (مانند ACE در اینجا)، همهٔ بسته‌ها از آن مسیر عبور خواهند کرد، حتی اگر شبکه چنین الزامی را تحمیل نکرده باشد.

تصمیم‌گیری دربارهٔ مسیر ارسال بسته‌ها امری داخلی است، یعنی هر مسیریاب خود دربارهٔ آن تصمیم می‌گیرد. وقتی یک بسته به مسیریاب A می‌رسد، این مسیریاب A است که تصمیم می‌گیرد آنرا از طریق خط متصل به B بفرستد یا از خط متصل به C . مسیریاب‌ها برای تصمیم‌گیری دربارهٔ مسیر بسته‌ها از **الگوریتمهای مسیریابی** (routing algorithm) استفاده می‌کنند، که دربارهٔ آنها در فصل ۵ صحبت خواهیم کرد.

تمام شبکه‌های WAN از نوع سوئیچ بسته نیستند، مانند سیستمهای ماهواره‌ای. در این سیستمها هر رواتر آنتنی دارد که از طریق آن اطلاعات را به ماهواره می‌فرستد، یا اطلاعات ارسالی آن را دریافت می‌کند. تمام مسیریاب‌های این مجموعه می‌توانند به ماهواره گوش کنند (و حتی برخی از آنها به اطلاعات ارسالی از مسیریاب‌های همسایه نیز گوش می‌کنند). البته شبکه‌هایی هم وجود دارد که فقط برخی از مسیریاب‌های آن (و نه همهٔ آنها) ارتباط ماهواره‌ای دارند. شبکه‌های ماهواره‌ای ذاتاً از نوع پخشی هستند، و اغلب در جاهایی بکار می‌روند که این طریقهٔ پخش اهمیت داشته باشد.

۱-۲-۴ شبکه‌های بیسیم (Wireless Network)

مخابرات دیجیتال بیسیم ایدهٔ جدیدی نیست. کُد مورسی که فیزیکدان ایتالیایی گاگلیلمو مارکونی در سال ۱۹۰۱ از یک کشتی به ساحل مخابره کرد، را می‌توان اولین پیام دیجیتال بیسیم محسوب کرد. سیستمهای جدید مخابرات

بیسیم فقط کارایی بهتری دارند، اما ایدهٔ اصلی در واقع همان است.

در ساده‌ترین صورت، شبکه‌های بیسیم را می‌توان به سه دستهٔ بزرگ تقسیم کرد:

۱. ارتباطات بین سیستمی
۲. LAN های بیسیم
۳. WAN های بیسیم

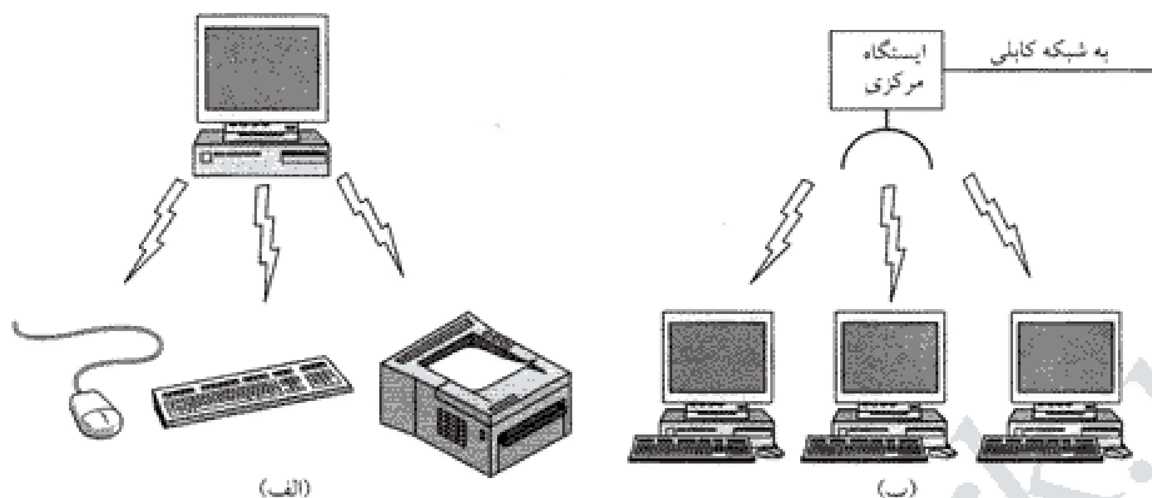
ارتباطات بین سیستمی (system interconnjection) یعنی برقراری ارتباط بین قطعات داخلی یک کامپیوتر با استفاده از امواج رادیویی کوتاه بُرد. تقریباً هر کامپیوتری یک مانیتور، صفحه کلید یا ماوس دارد که معمولاً با کابل به آن متصل می‌شوند. برای بسیاری از کاربران خانگی (و حتی اداری) وصل کردن این کابلها (با اینکه آنها طوری طراحی شده‌اند که نتوان هیچکدام را به دیگری وصل کرد) یک کار شاق است، و برای این کار دست به دامان تکنسین‌های کامپیوتر می‌شوند. به همین علت، برخی از شرکتهای سازندهٔ کامپیوتر دور هم جمع شدند، و یک شبکهٔ بیسیم با بُرد کوتاه بنام **بلوتوث** (Bluetooth) اختراع کردند که این قطعات را بدون استفاده از سیم به کامپیوتر متصل می‌کند. تکنولوژی بلوتوث اجازه می‌دهد تا دستگاههایی مانند چاپگر، دوربین دیجیتال، گوشی، و اسکنر نیز (با قرار گرفتن در بُرد امواج آن) به کامپیوتر متصل شوند. برای این کار به هیچ اتصال فیزیکی یا حتی نصب درایور نیاز نیست، و فقط کافیست دستگاه را روش کرده و در بُرد کامپیوتر قرار دهید، تا کار کند. برای بسیاری از کاربران این یک مزیت خارق‌العاده است.

ارتباطات بین سیستمی اساساً بر الگوی **اصلی-پیرو** (master-slave) مبتنی است (شکل ۱-۱۱ الف). در این سیستم، کامپیوتر اصلی است و با وسایل جانبی بعنوان رعایای خود صحبت می‌کند. این کامپیوتر اصلی است که به رعایا می‌گوید از چه آدرسی استفاده کنند، کی حرف بزنند، چه مدت حرف بزنند، روی چه فرکانسی صحبت کنند، و مانند آن. در بارهٔ تکنولوژی بلوتوث در فصل ۴ مفصلاً صحبت خواهیم کرد.

نوع دیگر ارتباطات بیسیم، شبکهٔ محلی بیسیم (یا LAN بیسیم) است. در این سیستم هر کامپیوتر یک مودم رادیویی و یک آنتن دارد، که به وسیلهٔ آن با کامپیوترهای دیگر ارتباط برقرار می‌کند. در اغلب این سیستمها یک آنتن مرکزی روی پشت بام وجود دارد (شکل ۱-۱۱ ب)، که ارتباط بین کامپیوترها را تسهیل می‌کند، اما اگر شبکه باندازهٔ کافی کوچک باشد، آنها می‌توانند مستقیماً با هم حرف بزنند. این نوع شبکه در دفاتر کوچک، خانه‌ها و جاهایی که کابل‌کشی مشکل است، بسرعت در حال گسترش است. مهمترین استاندارد LAN های بیسیم IEEE 802.11 نام دارد، که در اغلب سیستمها از آن استفاده می‌شود.

نوع سوم ارتباطات بیسیم، سیستمهای WAN بیسیم است. شبکهٔ رادیویی بکار رفته در سیستمهای تلفن همراه از این نوع است. این سیستمها اکنون نسل سوم خود را پشت سر می‌گذارند. نسل اول آنالوگ بود و فقط برای صدا از آن استفاده می‌شد. نسل دوم با اینکه دیجیتال شده بود، ولی باز هم فقط از صدا پشتیبانی می‌کرد. نسل سوم نیز دیجیتال است، و اینک همزمان از صدا و دیتا پشتیبانی می‌کند. WAN های بیسیم اساساً تفاوتی با LAN بیسیم ندارند، و فقط بُرد آنها بیشتر و البته نرخ انتقال داده‌ها کمتر است. LAN های بیسیم می‌توانند داده‌ها را با سرعتهایی در حد 50 Mbps (در محدودهٔ چند ده متر) منتقل کنند. نرخ انتقال داده‌ها در WAN های بیسیم بزحمت به 1 Mbps می‌رسد، ولی بُرد آنها بجای متر با کیلومتر سنجیده می‌شود. در فصل ۲ دربارهٔ این سیستمها بسیار خواهیم گفت.

علاوه بر این شبکه‌های کم سرعت، اکنون WAN های بیسیم پُرظرفیت نیز در دست توسعه است. دسترسی پُر سرعت به اینترنت از منزل و دفتر کار بدون استفاده از خطوط تلفن، از اولین کاربردهای این شبکه‌هاست. استاندارد این سیستم (که به آن سرویس توزیع چندنقطه‌ای محلی گفته می‌شود) IEEE 802.16 نام دارد، که دربارهٔ آن در فصل ۴ بیشتر صحبت خواهیم کرد.



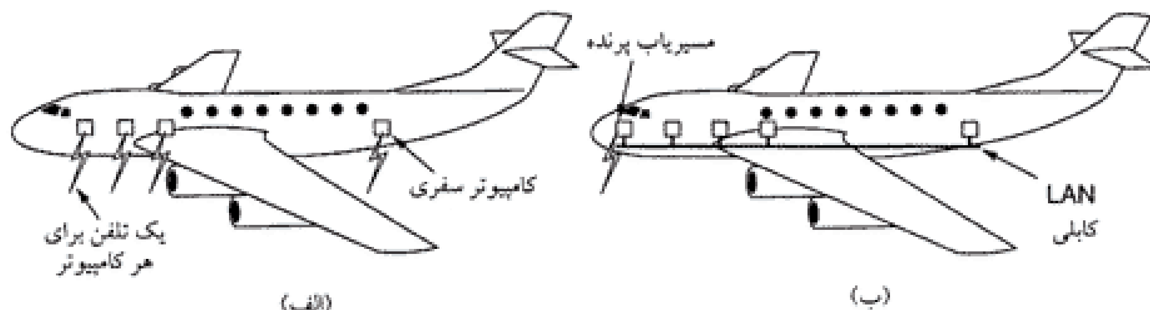
شکل ۱-۱۱. (الف) پیکربندی بلوتوث. (ب) LAN بیسیم.

تقریباً تمام شبکه‌های بیسیم باید در جایی به یک شبکهٔ معمولی متصل شوند، تا بتوانند وظایف خود را انجام دهند؛ این کار را به روشهای مختلف می‌توان انجام داد. برای مثال، در شکل ۱-۱۲ الف هواپیمایی را می‌بینید، که در آن تعدادی از مسافران مودمهای خود را به تلفنهای تعبیه شده در صندلی‌های هواپیما وصل کرده‌اند؛ این تماسها بکلی از هم مستقلند. اما در شکل ۱-۱۲ ب روش بهینه‌تری را ملاحظه می‌کنید، که در آن هر صندلی یک اتصال اترنت دارد، و مجموعهٔ آنها تشکیل یک LAN معمولی می‌دهند؛ و این LAN به یک مسیریاب بیسیم وصل است که ارتباط آنرا با دنیای خارج برقرار می‌کند.

بسیاری افراد فکر می‌کنند که بیسیم موج آینده است (برای مثال، Bi et al., 2001 ؛ Leeper, 2001 ؛ Varshey and Vetter, 2000)، ولی حداقل یک صدای مخالف هم وجود دارد. باب متکالف، مخترع اترنت، می‌گوید: «کامپیوترهای موبایل بیسیم مثل دستشویی‌های متحرک هستند - آنها فقط بدرد پیک نیک، کارگاه‌های ساختمانی و اردوهای کوتاه مدت می‌خورند. نصیحت من اینست که در خانهٔ خود کابل اترنت بکشید، و منتظر آینده بمانید» (Metcalfe, 1995). شاید تاریخ بعدها این اظهار نظر را در کنار جملهٔ تاریخی تی جی واتسون رئیس IBM در سال ۱۹۴۵ بگذارد، که در پاسخ اینکه چرا IBM وارد بازار کامپیوتر نمی‌شود، گفته بود: «دنیا تا سال ۲۰۰۰ به چهار یا پنج کامپیوتر بیشتر نیاز نخواهد داشت.»

۱-۲-۵ شبکه‌های خانگی (Home Network)

از هم اکنون می‌توان شبکه‌های خانگی را در افق آینده دید. ایدهٔ اصلی آن است که تمام یک وسایل یک خانه بتوانند



شکل ۱-۱۲. (الف) کامپیوترهای سفری منفرد. (ب) یک LAN پرنده.

با یکدیگر ارتباط برقرار کنند، و بتوان آنها را از طریق اینترنت کنترل کرد. این یکی از آن چیزهایی است که هیچکس منتظر آن نبوده (مانند، کنترل از راه دور تلویزیون و تلفن همراه)، ولی وقتی آمد دیگر هیچکس نمی‌تواند زندگی بدون آن را تصور کند.

وسایل زیادی را می‌توان در یک شبکه به هم متصل کرد، که از واضحترین آنها می‌توان به موارد زیر اشاره کرد:

۱. کامپیوترها (رومیزی، سفری، PDA ، وسایل جانبی)
۲. وسایل سرگرمی (تلویزیون، DVD ، ویدئو، دوربین دیجیتالی، استریو، MP3)
۳. وسایل مخابراتی (تلفن معمولی و همراه، فکس، دستگاههای ارتباط داخلی)
۴. لوازم خانگی (میکروپو، یخچال، ساعت، بخاری، تهویه مطبوع، چراغ)
۵. وسایل اندازه‌گیری از راه دور (آلارم دود یا دزدی، قرائت کنتور، ترموستات، دوربین اتاق بچه)

شبکه‌های خانگی به بسیاری از خانه‌ها راه یافته است؛ در این خانه‌ها وسایلی وجود دارد که یک ارتباط پُرسرعت اینترنت را بین چند کامپیوتر به اشتراک می‌گذارند. با گسترش سرگرمی‌ها روی اینترنت، بزودی شاهد تلویزیونها و استریوهایی خواهیم بود که مستقیماً به اینترنت متصلند (و این ارتباط دو جانبه خواهد بود، چرا که شاید شما هم مایل باشید فیلمها و موزیکهای خود را با دوستان و آشنایان به اشتراک بگذارید). مخابرات بین‌المللی از هم اکنون یک کالای در دسترس است، ولی بزودی این سرویسها بصورت دیجیتالی و از طریق اینترنت ارائه خواهند شد. امروزه کمتر خانه‌ای را می‌توان یافت که یک دوجین ساعت نداشته باشد، و با آمدن بهار و پائیز صاحبخانه مجبور است آنها را دستی جلو یا عقب بکشد؛ اگر تمام این ساعتها به اینترنت متصل باشند، می‌توان آنها را بصورت خودکار تنظیم کرد. کنترل خانه و مشاهدهٔ اتفاقاتی که در غیبت ما در آن می‌گذرد، یکی از آرزوهای دیرینهٔ ماست، که اینک به واقعیت تبدیل شده است. (دیگر می‌توانید با خیال راحت به سینما بروید، و بچه‌ها در خانه تنها بگذارید!) شاید فکر کنید هر یک از این کاربردها به شبکه‌ای مجزا نیاز دارد، اما یکپارچه کردن آنها احتمالاً ایدهٔ بهتری است.

شبکه‌های خانگی تفاوتهای ذاتی با سایر انواع شبکه دارد. اول اینکه نصب آن نباید پیچیده باشد. آنهایی که در این سالها درگیر کار نصب شبکه بوده‌اند، با جوابهای زیر (وقتی یک مشتری با مشکلی مواجه شده و به شما تلفن می‌زند) کاملاً آشنا هستند: (۱) دفترچهٔ راهنما را با دقت بخوانید، (۲) کامپیوتر را دوباره بوت کنید، (۳) تمام سخت‌افزارها و نرم‌افزارهای اضافی – آنهایی که مال شرکت ما نیست! – را حذف کنید، (۴) جدیدترین درایور را از سایت وب ما بردارید و نصب کنید، و بالاخره وقتی هیچکدام از این کارها فایده‌ای نبخشید، (۵) کامپیوتر را فرمت و ویندوز را از نو نصب کنید. اگر به کسی که یک یخچال اینترنتی خریده، بگوئید آخرین ویرایش سیستم عامل یخچال اینترنتی را بار کرده و نصب کند، مسلماً باعث خوشحالی وی نخواهد شد! آنهایی که کامپیوتر می‌خرند، عادت دارند با سیستمهایی سر و کله بزنند که کار نمی‌کند، ولی خریداران اتومبیل، تلویزیون و یخچال این حرفها سرشان نمی‌شود؛ آنها انتظار دارند وسیله‌ای که خریده‌اند از اول بسم‌الله بدون هیچ مشکلی کار کند.

دوم اینکه شبکه‌های خانگی باید بتوانند تحت هر شرایطی کار کنند. حتی همین حالا هم یک دستگاه کولر گازی با فقط چهار دکمهٔ OFF ، LOW ، MEDIUM و HIGH ، یک دفترچهٔ راهنمای ۳۰ صفحه‌ای دارد. تصورش را بکنید وقتی این کولر اینترنتی شود، فقط قسمت امنیت آن دستکم ۳۰ صفحه خواهد بود! و این از حد تحمل قسمت اعظم کاربران این قبیل دستگاهها خارج است.

قیمت پائین سومین عامل برای موفقیت شبکه‌های خانگی است. خیلی از افراد حاضر نیستند ۵۰ دلار پول اضافه برای یک ترموستات اینترنتی بدهند، که چی، فقط درجه حرارت خانهٔ خود را از اداره چک کنند! حتی ۵ دلار را هم شاید خیلی‌ها بزور بدهند.

از آنجائیکه دنیای آینده دنیای چندرسانه‌ایست، عامل چهارم در مقبولیت شبکه‌های خانگی بالا بودن پهنای باند آنهاست. مطمئن باشید هیچکس یک تلویزیون اینترنتی که فیلمها را با وضوح ۲۴۰×۳۲۰ پیکسل و ۱۰ فریم در ثانیه نشان می‌دهد، نخواهد خرید. حتی اترنت (که در اغلب ادارات از آن استفاده می‌شود) نیز مناسب چندرسانه‌ای نیست. شبکه‌های خانگی برای آن که بتوانند فراگیر شوند، باید پهنای باند بیشتر را با قیمتی کمتر ارائه کنند.

پنجم، امکان گسترش شبکه‌های خانگی است. این به معنای آن است که جنگ استانداردها باید یک بار و برای همیشه حل شود. اگر امروز به مشتریان خود توصیه کنید وسایل IEEE 1394 (که به **فایر وایر - FireWire -** معروف است) را بخرند، و سال آینده بگوئید امسال مد USB 2.0 است، مطمئن باشید همهٔ آنها را پراتده‌اید. استانداردهای ارتباطی باید سالها (و کابل‌کشی، دهها سال) ثابت و بدون تغییر باقی بماند.

امنیت و قابلیت اعتماد ششمین عامل موفقیت شبکه‌های خانگی است. اینکه چند تا ایمیل ویروسی از دست بدهید یک چیز است، و به باد رفتن تمام خانه در اثر لو رفتن کُدهای امنیتی آن، یک چیز دیگر.

در این میان یکی از سؤالات جالب اینست که شبکه‌های خانگی باید بیسیم باشند یا با سیم. همین حالا هم در اغلب خانه‌ها شش شبکه وجود دارد: شبکهٔ برق، تلفن، تلویزیون کابلی، آب، گاز و فاضلاب. اضافه کردن شبکهٔ هفتم به خانه‌های نوساز هزینهٔ چندانی ندارد، ولی در خانه‌های موجود کاری پُرخرج خواهد بود. شبکه‌های بیسیم کم خرج‌تر هستند، ولی شبکه‌های کابلی از نظر ایمنی بسیار قابل اعتمادترند. امواج رادیویی بسادگی از محدودهٔ خانه خارج می‌شوند، و شاید یک همسایهٔ فضول بتواند ایمیل‌های شما را بخواند، و یا دزدکی از اشتراک اینترنت شما استفاده کند. در فصل ۸ دربارهٔ روشهای رمزنگاری برای مقابله با این مشکلات صحبت خواهیم کرد، ولی در شبکه‌های خانگی موضوع امنیت از اهمیت چندگانه‌ای برخوردار است.

بطور خلاصه، شبکه‌های خانگی علاوه بر امکاناتی که ارائه می‌کنند، چالشهایی را نیز با خود به همراه دارند. اغلب این چالشها به مدیریت ساده، قابلیت اعتماد، ایمنی بالا (بویژه در مورد کاربران غیرحرفه‌ای)، کارایی بالا و قیمت پائین مربوط می‌شوند.

۱-۲-۶　شبکهٔ شبکه‌ها (Internetwork)

شبکه‌های متعددی با نرم‌افزارها و سخت‌افزارهای بسیار مختلف در سراسر دنیا وجود دارد، و بسیار پیش می‌آید که کاربری از یک شبکه بخواهد با کاربران شبکه‌های دیگر ارتباط برقرار کند. برای انجام این خواسته بایستی شبکه‌های مختلف (که بعضاً با هم ناسازگار هم هستند) با وسایلی بنام **دروازه** gateway - که می‌تواند سخت‌افزاری یا نرم‌افزاری باشد - به هم متصل شده، و داده‌ها از فرمتی به فرمت دیگر تبدیل شود. به مجموعه‌ای از این شبکه‌های بهم پیوسته **شبکهٔ شبکه‌ها** (internetwork یا internet) گفته می‌شود. کلمهٔ internet وقتی با i نوشته می‌شود، معنای عام می‌دهد، ولی با I همان شبکهٔ جهانی اینترنت از آن مستفاد می‌شود.

متداولترین شکل شبکهٔ شبکه‌ها عبارتست از تعدادی LAN که با ارتباطات WAN به هم متصل شده‌اند. در حقیقت، اگر در شکل ۱-۹ WAN را به جای subnet (زیرشبکه) قرار دهیم، هیچ چیز تغییر نخواهد کرد. در این مورد تنها تفاوت تکنیکی بین WAN و subnet وجود یا عدم وجود کامپیوترهای میزبان (host) است: اگر سیستم ناحیهٔ خاکستری فقط از مسیریاب تشکیل شده باشد، این یک زیرشبکه است؛ اگر در آن میزبان هم وجود داشته باشد، یک WAN است. تفاوت واقعی در مالکیت و طرز استفاده است.

اغلب افراد مفاهیم **زیرشبکه** (subnet)، **شبکه** (network) و **شبکهٔ شبکه‌ها** (internetwork) را با هم اشتباه می‌کنند. زیرشبکه (که بیشتر در شبکه‌های گسترده مفهوم پیدا می‌کند) مجموعه‌ایست از مسیریاب‌ها و خطوط مخابراتی متعلق به راهبر شبکه. برای مقایسه، شبکهٔ تلفن شهری از یک سری مراکز سوئیچینگ، خطوط مخابراتی

پُرسرعت، و خطوط کم سرعت که مشترکان را به مرکز وصل می‌کند، تشکیل می‌شود. خطوط پُرسرعت و مراکز سوئیچینگ (که متعلق به شرکت مخابرات هستند) همان **زیرشبکه** را می‌سازند. تلفنهای مشترکان که متعلق به افراد (**و معادل‌میزبان**) است، جزیی از زیرشبکه نیستند. مجموعهٔ این دو (زیرشبکه و میزبانها) **شبکه** را می‌سازد. در شبکه‌های LAN (که فقط کامپیوتر و کابل است) زیرشبکه وجود ندارد.

وقتی چند تا از این شبکه‌ها به هم متصل می‌شوند، یک **شبکهٔ شبکه‌ها** شکل می‌گیرد. با این تعریف، مجموعهٔ دو LAN ، یا یک LAN و یک WAN ، را هم می‌توان شبکهٔ شبکه‌ها نامید، ولی در صنعت کامپیوتر توافق یکپارچه‌ای دربارهٔ این اصطلاحات وجود ندارد. بعنوان یک قاعدهٔ کلی، اگر چند شرکت یا سازمان مختلف هر یک (به هزینهٔ خود) قسمتی از یک شبکه را بنا کنند، با یک شبکهٔ شبکه‌ها سروکار داریم. همچنین اگر تکنولوژی زیربنایی در قسمتهای مختلف متفاوت باشد، به احتمال زیاد دو شبکه داریم.

۳ـ۱ نرم‌افزار شبکه

در اولین شبکه‌های کامپیوتری سخت‌افزار از اهمیت ویژه‌ای برخوردار بود، و به نرم‌افزار فقط بعنوان چیزی که باید بعداً به آن فکر می‌شد، نگاه می‌کردند. اما این استراتژی دیگر کارایی ندارد. امروزه نرم‌افزار شبکه بسیار ساخت‌یافته است، که در این قسمت آنرا بررسی می‌کنیم. روشهای مورد بحث در این قسمت سنگ بنای کتاب را تشکیل می‌دهند، و در آینده بسیار به آنها مراجعه خواهیم کرد.

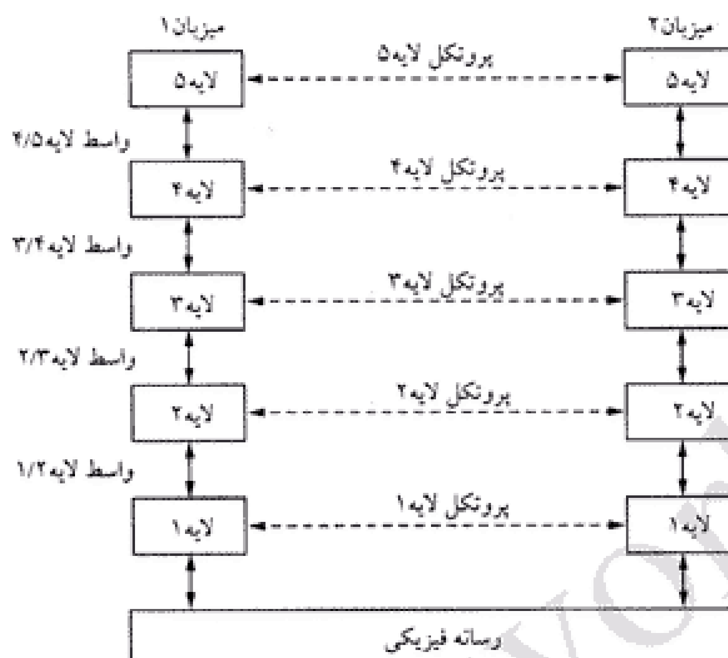
۱ـ۳ـ۱ سلسله مراتب پروتکل‌ها

برای کاهش پیچیدگیهای طراحی، اغلب شبکه‌ها بصورت مجموعه‌ای از چند **لایه** (layer) یا **سطح** (level) – که هر کدام روی دیگری قرار می‌گیرند – طراحی می‌شوند. تعداد لایه‌ها، نام هر لایه، محتوای آن، و کاری که هر لایه انجام می‌دهد، از شبکه‌ای به شبکهٔ دیگر متفاوت است. وظیفهٔ هر لایه ارائهٔ سرویسهای خاص به لایه‌های بالاتر، و پنهان کردن جزئیات کار از دید آنهاست. در این مفهوم، هر لایه یک **ماشین مجازی** (virtual machine) است که سرویسهای خاصی را در اختیار لایه‌های بالاتر می‌گذارد.

این یکی از مفاهیم آشنا و کلیدی در کلیهٔ علوم کامپیوتری است، که با نامهایی از قبیل **پنهان کردن اطلاعات** (information hiding) ، **انواع دادهٔ مجرد** (abstract data types) ، **کپسولی کردن داده‌ها** (data encapsulation) و **برنامه‌نویسی شیء‌گرا** (object-oriented programming) شناخته می‌شود. ایدهٔ اصلی این است که یک قطعه نرم‌افزار (یا سخت‌افزار) سرویسی را به کاربران خود عرضه کند، ولی جزئیات کار (از قبیل حالت داخلی خود و الگوریتمهای بکار رفته) را از آنها مخفی نگه دارد.

لایهٔ *n* یک ماشین همیشه با لایهٔ *n* ماشین دیگر حرف می‌زند. قواعد و قراردادهای این ارتباط را پروتکل لایهٔ *n* (layer *n* protocol) می‌نامند. در ساده‌ترین حالت، **پروتکل** (protocol) عبارتست از قراردادهای توافق شده بین دو طرف برای برقراری و پیشبرد یک ارتباط. بعنوان مقایسه، وقتی یک خانم به یک آقا معرفی می‌شود، آن خانم می‌تواند دستش را پیش بیاورد؛ و بنوبهٔ خود آن آقا هم می‌تواند دست خانم را (اگر یک شاهزادهٔ اروپایی در یک میهمانی رسمی باشد) ببوسد، یا فقط با او دست بدهد (اگر آن خانم یک وکیل امریکایی در یک جلسهٔ کاری باشد). سرپیچی از پروتکل‌ها برقراری ارتباط را بسیار دشوار (اگر نگوئیم، غیرممکن) خواهد کرد.

در شکل ۱–۱۳ یک شبکه پنج لایه به تصویر کشیده شده است. به اجزایی که در یک لایه هستند، **همتا** (peer) گفته می‌شود. این همتاها می‌توانند پروسس‌های نرم‌افزاری، وسایل سخت‌افزاری، و یا حتی دو انسان باشند. بعبارت دیگر، این همتاها هستند که با استفاده از پروتکل با هم رابطه برقرار می‌کنند.



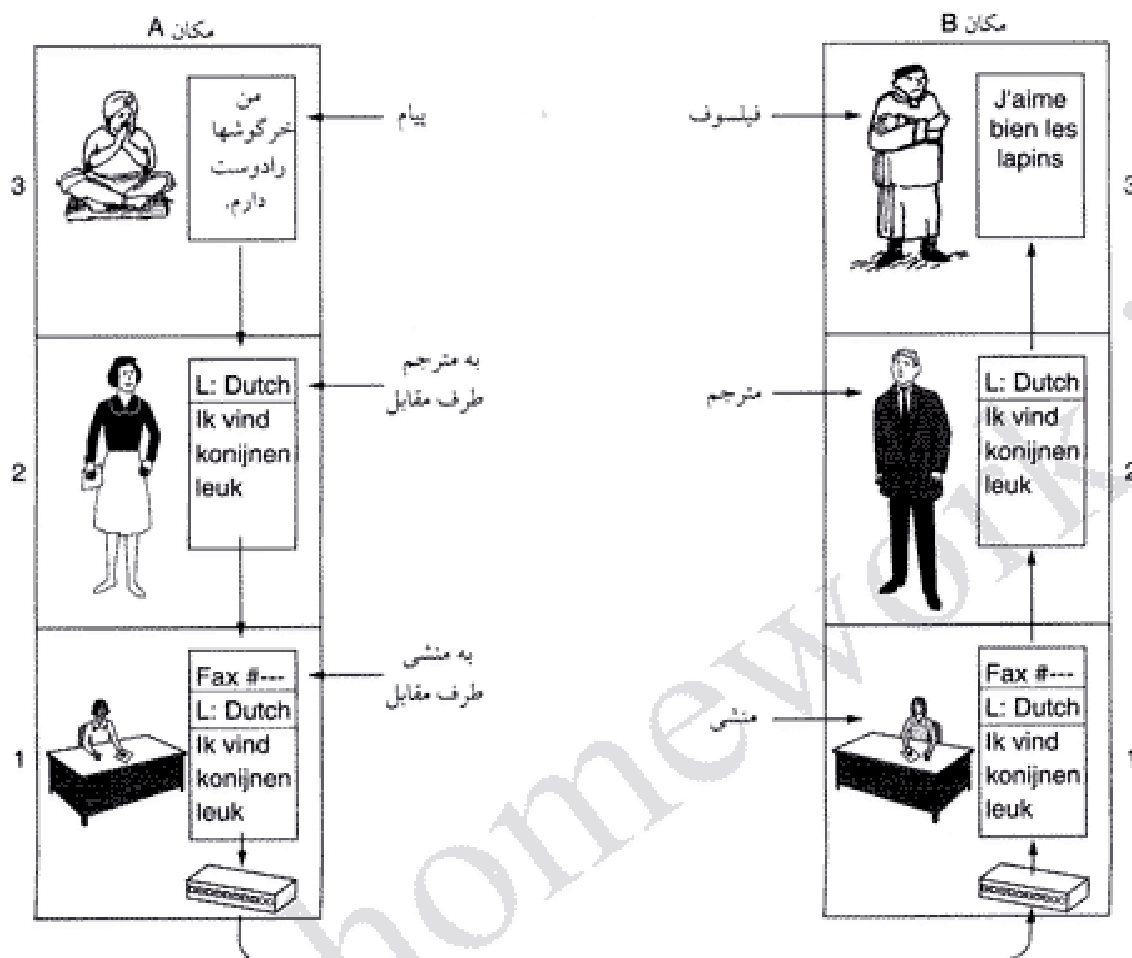
شکل ۱-۱۳. لایه‌ها، پروتکل‌ها، و واسط‌ها.

در حقیقت، داده‌ها هرگز مستقیماً از لایهٔ n یک ماشین به لایهٔ n ماشین دیگر منتقل نمی‌شوند. بلکه، هر لایه داده‌ها (و اطلاعات کنترلی) را به لایهٔ زیرین خود می‌دهد، تا به پائین‌ترین لایه برسد. در زیر پائینترین لایه (لایهٔ 1) **رسانهٔ فیزیکی** (physical medium) قرار دارد، که داده‌ها را جابجا می‌کند. در شکل ۱-۱۳ ارتباط مجازی لایه‌ها با خط‌چین و ارتباط واقعی و فیزیکی با خط ممتد نشان داده شده است.

بین هر زوج از لایه‌های مجاور **واسط** (interface) قرار دارد. واسط مشخص می‌کند که هر لایه چه سرویسها و عملکردهای پایه‌ای در اختیار لایهٔ بالاتر می‌گذارد. تعریف واسط‌های مناسب از مهمترین وظایف طراحان شبکه است. لازمهٔ این امر آنست که وظایف هر لایه دقیقاً مشخص و شناخته شده باشد. علاوه بر به حداقل رساندن اطلاعات رد و بدل شده بین لایه‌ها، یک واسط شسته و رفته کار عوض کردن پیاده‌سازی لایه‌ها را نیز آسان می‌کند، چون تنها کاری که باید کرد این است که پیاده‌سازی جدید دقیقاً همان سرویسهای پیاده‌سازی قدیمی را به همسایگان خود ارائه کند. در حقیقت، پیاده‌سازیهای متعددی در شبکه‌های مختلف وجود دارد، که هیچ خللی در ارتباط لایه‌ها ایجاد نمی‌کنند.

به مجموعهٔ لایه‌ها و پروتکل‌ها **معماری شبکه** (network architecture) می‌گویند. مشخصه‌های یک معماری باید آنچنان دقیق و جامع باشد که طراح شبکه بتواند نرم‌افزارها و سخت‌افزارهای لازم برای کارکرد صحیح آنرا فراهم آورد. جزئیات پیاده‌سازی و مشخصات واسطها هرگز جزء معماری شبکه نیست، چون آنها باید در دل ماشین مخفی باشند (و از خارج دیده نشوند). حتی لازم نیست واسطها در تمام ماشینهای یک شبکه یکسان باشند، مشروط باینکه تمام این ماشینها بتوانند از تمام پروتکل‌ها استفاده کنند. به مجموعه پروتکلهایی که در یک سیستم خاص بکار می‌روند (یک پروتکل در هر لایه)، **پشته پروتکل** (protocol stack) گفته می‌شود. پروتکل، پشته پروتکل و معماری شبکه از مهمترین موضوعات مورد بحث این کتاب هستند.

شاید یک مثال بتواند در روشن شدن مفهوم ارتباط چند لایه کمک کند. دو فیلسوف را، که اولی فقط زبانهای اردو و انگلیسی می‌داند و دیگری فقط چینی و فرانسه، در نظر بگیرید (آنها معادل پروسسهای همتا در لایهٔ 3 هستند). از آنجائیکه این دو فیلسوف نمی‌توانند مستقیماً با هم حرف بزنند، دو مترجم استخدام می‌کنند



شکل ۱-۱۴. معماری فیلسوف-مترجم-منشی.

(پروسسهای همتا در لایهٔ 2)، که آنها هم بنوبهٔ خود هر کدام یک منشی دارند (پروسسهای همتا در لایهٔ 1). فیلسوف 1 میل دارد علاقهٔ خود به *oryctolagus cuniculus* را به فیلسوف ۲ (همتای خود) ابلاغ کند. برای اینکار، از طریق واسط 2/3 یک پیام بزبان انگلیسی با مضمون "I like rabbits" به مترجم خود می‌فرستد (شکل ۱-۱۴ را ببینید). مترجم‌ها بین خود توافق کرده‌اند که به زبان هلندی حرف بزنند، پس پیام فوق به "Ik vind konijnen leuk" تبدیل می‌شود. انتخاب زبانی که همتاهای لایهٔ 2 با آن صحبت کنند، بر عهدهٔ خود آنان (پروسسهای این لایه) است.

مترجم، سپس، این پیام را به منشی خود می‌دهد تا مثلاً از طریق فکس (پروتکل لایهٔ 1) ارسال شود. وقتی این پیام به منشی سمت مقابل رسید، آنرا به مترجم (لایهٔ 2) تحویل داده، و مترجم نیز پس از ترجمه به زبان فرانسه آنرا، از طریق واسط 2/3 ، به فیلسوف 2 می‌دهد. توجه کنید که تا زمانیکه واسطها تغییر نکرده باشند، پروتکل‌ها کاملاً از یکدیگر مستقل هستند. برای مثال، مترجمها می‌توانند زبان توافقی خود را به فنلاندی تغییر دهند، مشروط باینکه واسط آنها با لایه‌های 1 و 3 هیچ تغییری نکند. یا اینکه منشی‌ها می‌توانند از فکس به ایمیل یا تلفن سوئیچ کنند، بدون اینکه این کار هیچ تأثیری روی لایه‌های دیگر بگذارد. هر پروسس می‌تواند اطلاعات دلخواه خود را (که فقط بدرد پروسس همتای آن می‌خورد) به پیام اضافه کند. این اطلاعات به لایه‌های بالاتر منتقل نخواهند شد.

حال یک مثال فنی‌تر را در نظر می‌گیریم: نحوهٔ ارتباط در یک شبکهٔ پنج لایه (شکل ۱-۱۵ را ببینید). برنامه‌ای

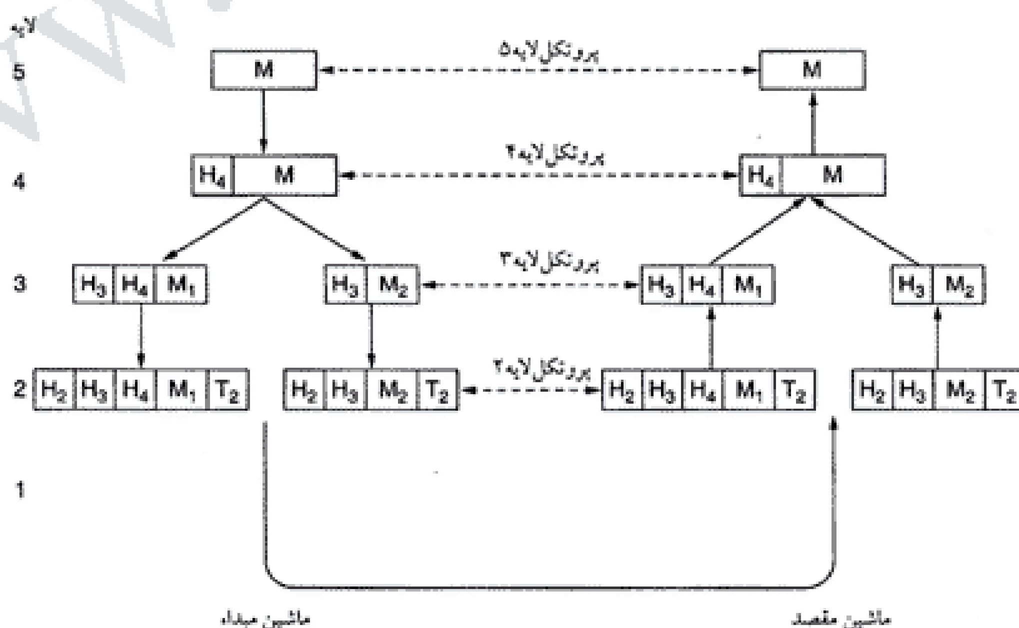
که در لایهٔ 5 اجرا می‌شود، یک پیام *(M)* تولید کرده و برای ارسال به لایهٔ 4 می‌دهد. لایهٔ 4 یک **سرآیند** (header) به این پیام اضافه، و آنرا به لایهٔ 3 تحویل می‌دهد. سرآیند حاوی اطلاعات کنترلی بین لایه‌های متناظر است؛ برای مثال، لایهٔ 4 می‌تواند به هر پیام یک عدد ترتیبی نسبت بدهد، تا اگر لایه‌های پائینتر آنها را بدون نظم و ترتیب ارسال کردند، لایهٔ متناظر در سمت مقابل بتواند آنها را بترتیب صحیح بازیابی کند. در برخی از لایه‌ها، این سرآیندها حاوی اندازهٔ بسته، زمان ارسال و اطلاعاتی از این قبیل است.

در بسیاری از شبکه اندازهٔ پیام در لایهٔ 4 هیچ محدودیتی ندارد، ولی (تقریباً همیشه) این محدودیت در لایهٔ 3 وجود دارد. در نتیجه، لایهٔ 3 باید پیام را به قطعات کوچکتر بشکند، و به هر قطعه یک سرآیند لایهٔ 3 اضافه کند. در این مثال، پیام M به دو قطعهٔ M_1 و M_2 شکسته شده است.

سپس، لایهٔ 3 این بسته‌ها را به لایهٔ بعدی (یعنی لایهٔ 2) تحویل می‌دهد. لایهٔ 2 علاوه بر اضافه کردن سرآیند خاص خود به ابتدای هر بسته، به انتهای آنها نیز یک **دنباله** (trailer) چسبانده، و آنها را به لایهٔ بعدی (که لایهٔ انتقال فیزیکی است) می‌دهد. در سمت مقابل، ماشینی که این بسته‌ها را دریافت می‌کند، آنها را لایه به لایه بالا می‌فرستد، و هر لایه (قبل از تحویل به لایهٔ بالاتر) اطلاعات خاص خود را از بسته‌ها برمی‌دارد. بدین‌ترتیب، هیچ لایه‌ای سرآیندها و دنباله‌های لایه‌های زیرین خود را دریافت نخواهد کرد.

نکتهٔ مهمی که در شکل ۱-۱۵ باید بدان توجه کنید، رابطهٔ ارتباطات مجازی و حقیقی بین ماشینها، و تفاوت پروتکل‌ها و واسط‌هاست. برای مثال، پروسسهای همتا در لایهٔ 4 فکر می‌کنند که ارتباط بین آنها یک ارتباط «افقی» است، که با استفاده از پروتکل لایهٔ 4 برقرار می‌شود. هر یک از این پروسس‌ها احتمالاً ابزارهایی بنام *SendToOtherSide* (برای ارسال پیام به سمت مقابل) یا *GetFromOtherSide* (برای دریافت پیام از سمت مقابل) نیز دارند، و حتی نمی‌دانند که پیامهای خود را از طریق واسط 3/4 به لایهٔ پائین‌تر می‌دهند، نه به طرف مقابل.

تجریدی بودن رابطهٔ پروسسهای همتا یک نکتهٔ کلیدی در طراحی شبکه است. با این تمهید، کار بسیار پیچیده و دشواری مانند طراحی کامل یک شبکه به کارهای کوچکتر و ساده‌تری (مانند طراحی لایه‌های جداگانه) شکسته می‌شود.



شکل ۱-۱۵. انتقال اطلاعات در یک شبکهٔ پنج لایه.

با اینکه نام این بخش «نرم‌افزار شبکه» است، اما باید متوجه باشید که لایه‌های پائین‌تر معمولاً بصورت سخت‌افزاری پیاده‌سازی می‌شوند. با این حال، اینها نیز الگوریتمهای پیچیدهٔ نرم‌افزاری هستند، که فقط در سخت‌افزار حک و ثابت شده‌اند.

۱-۳-۲ ملاحظاتی در طراحی لایه‌ها

برخی از مفاهیم کلیدی طراحی شبکه‌های کامپیوتری در لایه‌های مختلفی حضور دارند، که در زیر برخی از مهمترین آنها را بطور مختصر بررسی خواهیم کرد.

هر لایه به مکانیزمی برای شناسایی **فرستنده** (sender) و **گیرنده** (receiver) نیاز دارد. از آنجائیکه یک شبکه معمولاً تعداد زیادی کامپیوتر دارد، و در هر کامپیوتر پروسس‌های متعددی در حال اجرا هستند، باید ابزاری وجود داشته باشد که هر پروسس بتواند پروسس همتای خود را دقیقاً شناسایی و مشخص کند. بعبارت دیگر، برای تعیین دقیق مقصد به یک نظام **آدرس‌دهی** (addressing) نیاز داریم.

نحوهٔ انتقال داده‌ها نیز نیازمند قواعد و مقررات خاص خود است. در برخی از سیستمها داده‌ها فقط در یک جهت حرکت می‌کنند، اما در برخی دیگر در هر دو جهت. همچنین پروتکل باید تعیین کند که هر ارتباط فیزیکی معادل چند کانال منطقی است، و اولویت‌بندی آنها چگونه است. در بسیاری از شبکه‌ها هر ارتباط فیزیکی حداقل دو کانال منطقی را شامل می‌شود، یکی برای داده‌های معمولی و دیگری برای داده‌های اضطراری.

مسئلهٔ مهم دیگر **کنترل خطا** (error control) است، چون هیچ ارتباط فیزیکی صد در صد کامل و عاری از خطا نیست. کُدهای بسیاری برای کشف و تصحیح خطا وجود دارد، ولی هر دو طرف باید بر سر یکی از آنها توافق کنند. همچنین گیرنده باید بتواند به طریقی به فرستنده اعلام کند که کدام پیامها را درست دریافت کرده و کدامها را غلط.

همانطور که قبلاً هم گفتیم، در بسیاری از موارد بسته‌های تشکیل‌دهندهٔ یک پیام بصورت منظم ارسال نمی‌شوند؛ در این حالت، پروتکل باید طوری طراحی شده باشد که گیرنده بتواند قطعات پیام را دوباره بنحو صحیح بهم بچسباند. یک راه حل شماره‌گذاری قطعات (بسته‌های) پیام است، ولی باز این سؤال باقیست که گیرنده باید چگونه با این بسته‌ها عمل کند.

یکی از از مسائلی که در تمام لایه‌ها وجود دارد اینست که، فرستنده با چه سرعتی باید اطلاعات را ارسال کند تا گیرنده‌های کُند در گرداب داده‌ها غرق نشوند. برای این مشکل نیز راه حل‌های مختلفی وجود دارد که بعداً مفصلاً دربارهٔ آنها توضیح خواهیم داد. برخی از این راه حل‌ها شامل نوعی فیدبک (بازخور) از گیرنده به فرستنده است، که در هر لحظه وضعیت گیرنده را (بطور مستقیم یا غیرمستقیم) به فرستنده اعلام می‌کند. در برخی دیگر، دو طرف از قبل بر سر نرخ انتقال اطلاعات توافق می‌کنند. این مبحث **کنترل جریان** (flow control) نام دارد.

مسئلهٔ دیگری که در لایه‌های متعدد باید حل شود اینست که، اغلب پروسسها قادر نیستند پیامهایی با هر طول دلخواه دریافت کنند. این وضعیت باعث شده تا مکانیزمهایی برای شکستن پیامها به قطعات کوچک، ارسال، و سپس مونتاژ آنها در مقصد ابداع شود. مشکل دیگری که از اینجا ناشی می‌شود آنست که برخی از پروسسها اصرار دارند پیامها را آنقدر ریز کنند، بگونه‌ایکه کارایی کل سیستم را مختل می‌کند. در اینجا راه حل چسباندن چند قطعه به یکدیگر و شکستن دوبارهٔ آنها در مقصد است.

در اغلب موارد ایجاد کانالهای ارتباطی جداگانه برای هر زوج پروسس کاری پُرهزینه (و گاهی غیرممکن) است. در این موارد، لایه‌های پائین‌تر برای برقراری ارتباط بین چند پروسس مستقل، از یک کانال استفاده می‌کنند. این عمل (که **مالتی‌پلکس** – multiplexing – و **دمالتی‌پلکس** – demultiplexing – نام دارد) معمولاً بصورت شفاف انجام می‌شود، که در اینصورت می‌توان آنرا در هر لایه‌ای پیاده‌سازی کرد. برای مثال، وقتی تعداد خطوط

مخابراتی موجود محدود است، در لایهٔ فیزیکی از تکنیکهای مالتی‌پلکس استفاده می‌شود.

وقتی بین مبدأ و مقصد مسیرهای مختلفی وجود دارد، یکی از آنها باید انتخاب شود. گاهی تصمیم‌گیری در این مورد باید در چند لایه انجام شود. مثلاً، برای ارسال اطلاعات از لندن به رُم اولاً (در لایه‌های بالاتر) باید تصمیم بگیریم که از مسیر فرانسه استفاده کنیم یا از مسیر آلمان؛ سپس در لایه‌های پائینتر انتخاب مسیرهای خلوت‌تر از میان مسیرهای موجود پیش می‌آید. به این مبحث **مسیریابی** (routing) گفته می‌شود.

۱-۳-۳ سرویسهای اتصال-گرا و غیرمتصل

هر لایه می‌تواند دو نوع سرویس در اختیار لایهٔ بالاتر از خود بگذارد: **سرویس اتصال-گرا** (connection-oriented) و **سرویس غیرمتصل** (connectionless) . در این قسمت این سرویس‌ها و تفاوتهای آنها را بررسی خواهیم کرد.

سرویس اتصال-گرا بر اساس مدل سیستمهای تلفن کار می‌کند. وقتی می‌خواهید با یک نفر تماس بگیرید، گوشی تلفن را برداشته، شماره می‌گیرید، صحبت می‌کنید، و بعد گوشی را می‌گذارید. در یک سرویس اتصال-گرا هم ابتدا اتصال برقرار شده، و بعد از تبادل اطلاعات موردنظر، اتصال قطع می‌شود. مهمترین نکته در مورد سرویسهای اتصال-گرا اینست که آنها مانند یک لوله عمل می‌کنند: فرستنده از یک طرف داده‌ها (بیت‌ها) را به داخل لوله می‌فرستند، و گیرنده در طرف دیگر آنها را می‌گیرد. در اغلب موارد داده‌ها بهمان ترتیبی که فرستاده شده‌اند، دریافت می‌شوند.

در برخی موارد بعد از برقراری اتصال، فرستنده، گیرنده و زیرشبکه ابتدا یک سری **مذاکرات اولیه** (negotiation) انجام می‌دهند تا بر سر مواردی از قبیل حداکثر اندازهٔ پیامها، کیفیت سرویس موردنظر، و مانند آن توافق کنند. معمولاً، یک طرف پیشنهادی می‌دهد و طرفهای دیگر آنرا قبول یا رد کرده، و یا بکلی پیشنهاد جدیدی ارائه می‌کنند.

از سوی دیگر، سرویس غیرمتصل بر اساس مدل پُست بنا شده است. هر پیام (نامه) دارای آدرس مشخصی است، و مسیری که برای رسیدن به مقصد طی می‌کند، کاملاً مستقل از پیامهای دیگر است. معمولاً وقتی دو نامه به یک مقصد می‌فرستید، اولین نامه زودتر از دومی به آنجا می‌رسد؛ ولی گاهی پیش می‌آید که اولی با تأخیر و بعد از دومی به مقصد برسد.

برای هر سرویس می‌توان یک **کیفیت سرویس** (quality of service) در نظر گرفت. برخی از سرویسها مطمئن و قابل اعتماد هستند، بگونه‌ایکه هیچ داده‌ای در حین انتقال از بین نمی‌رود. یک سرویس قابل اعتماد معمولاً بگونه‌ای طراحی می‌شود که گیرنده دریافت صحیح داده‌ها را به فرستنده اعلام کند. این **تصدیق دریافت** (acknowledgement) باعث تحمیل یک بار اضافی و تأخیر در انتقال پیامها می‌شود، که اغلب ارزش آنرا دارد، ولی گاهی به زحمتش نمی‌ارزد.

انتقال فایل (file transfer) از جمله مواردیست که به یک سرویس مطمئن اتصال-گرا نیاز دارد؛ صاحب فایل معمولاً میل دارد تمام بیت‌های فایلش (با همان نظم و ترتیب) به مقصد برسد. کمتر کسی را پیدا می‌کنید که راضی شود یک فایل قروقاطی دریافت کند، حتی اگر اینکار به معنای سرعت بیشتر باشد.

سرویس اتصال-گرای قابل اعتماد بر دو گونهٔ مختلف است: **توالی پیام** (message sequence) و **جریان بایت** (byte stream) . در سرویس توالی پیام حد و مرز پیامها همیشه حفظ می‌شود: وقتی دو پیام ۱۰۲۴ بایتی می‌فرستید، طرف مقابل همیشه دو پیام ۱۰۲۴ بایتی دریافت خواهد کرد، نه یک پیام ۲۰۴۸ بایتی، یا چهار پیام ۵۱۲ بایتی. اما در سرویس دوم، چیزی بنام حد و مرز پیام وجود ندارد، و فقط جریانی از بایتها دیده خواهد شد. در این حالت وقتی ۲۰۴۸ بایت به مقصد می‌رسد، به هیچ طریقی نمی‌توان گفت که آیا این یک پیام ۲۰۴۸ بایتی بوده، یا دو

پیام ۱۰۲۴ بایتی، و یا ۲۰۴۸ پیام ۱ بایتی. برای مثال، اگر بخواهید صفحات یک کتاب را به یک دستگاه حروفچینی الکترونیکی بفرستید، شاید برایتان مهم باشد که حد و مرز هر صفحه مشخص باشد. از طرف دیگر، وقتی از راه دور به یک کامپیوتر متصل می‌شوید، جریان بایت‌ها از کامپیوتر مبدأ به مقصد تمام آن چیزیست که نیاز دارید، و حد و مرز پیامها هیچ اهمیتی ندارد.

اما همانطور که گفتیم، در برخی از موارد تأخیری که در نتیجهٔ تصدیق دریافت (acknowledgement) پدید می‌آید، غیر قابل قبول است؛ مکالمهٔ دیجیتالی یکی از این موارد است. اغلب کاربران ترجیح می‌دهند گاهی صدای طرف مقابل را با نویز بشنوند، تا اینکه (در نتیجهٔ مکانیزم تصدیق دریافت) مکالمه با تأخیر و وقفه انجام شود. و یا در کنفرانسهای ویدئویی کمی برفک و نویز قابل تحمل‌تر است، تا اینکه تصاویر با پرشهای اعصاب خردکن دریافت شود.

از طرف دیگر، همهٔ کاربردها به ارتباط متصل نیاز ندارند؛ پُست الکترونیک (ایمیل) یکی از مواردیست که نیازی به سرویس اتصال-گرا ندارد، و بویژه در مواقعی که هزینه عاملی تعیین‌کننده است، سرویس قابل اعتماد نیز چندان الزامی نیست. در این موارد فقط کافیست اولویت تحویل پیام بالا باشد. سرویس غیرمتصل غیرقابل‌اعتماد (سرویسی که به تصدیق دریافت از طرف مقابل متکی نیست) اغلب بعلت شباهتی که با سیستم تلگراف دارد، **سرویس دیتاگرام** (datagram service) نامیده می‌شود.

اما گاهی با اینکه نیازی به برقراری یک اتصال نیست (مثلاً برای ارسال یک پیام کوتاه)، ولی قابل‌اعتماد بودن ارتباط اهمیت دارد. در این قبیل موارد می‌توان از **سرویس دیتاگرام همراه با تصدیق دریافت** (acknowledged datagram service) استفاده کرد. این مانند پُست کردن یک نامهٔ سفارشی است، که فرستنده می‌خواهد از رسیدن نامه بدست گیرنده مطمئن شود. با دریافت این تأییدیه، فرستنده مطمئن می‌شود که نامه گم نشده و بدست طرف مقابل رسیده است.

سرویس دیگری نیز وجود دارد، که **سرویس درخواست-پاسخ** (request-reply service) نام دارد. در این سرویس، فرستنده دیتاگرامی که حاوی یک درخواست است ارسال می‌کند، و پاسخ آنرا می‌گیرد. برای مثال، وقتی پیامی به کتابخانهٔ محلی می‌فرستید و سؤال می‌کنید که «زبان سواحیلی در کدام کشور تکلم می‌شود»، از این سرویس استفاده می‌کنید. از سرویس درخواست-پاسخ معمولاً در سیستمهای مشتری-سرویس‌دهنده استفاده می‌شود: مشتری درخواست خود را به سرویس‌دهنده فرستاده، و پاسخ آنرا دریافت می‌کند. در شکل ۱-۱۶ خلاصه‌ای از سرویسهای توضیح داده شده در این قسمت را مشاهده می‌کنید.

	سرویس	مثال
اتصال گرا	استریم پیام قابل‌اعتماد	چندصفحه متوالی
	استریم بایت قابل‌اعتماد	ورود از راه دور
	اتصال غیرقابل‌اعتماد	صدای دیجیتالی
غیرمتصل	دیتا گرام غیرقابل‌اعتماد	
	دیتا گرام تصدیق شده	ایمیل ثبت شده
	درخواست - پاسخ	جستجوی پایگاه داده

شکل ۱-۱۶. شش نوع سرویس مختلف.

شاید در نگاه اول تعجب کنید که اصولاً چرا باید از یک سرویس غیرقابل‌اعتماد استفاده کنیم، و اصلاً چه کسی چنین چیزی را لازم دارد؟ اول از همه اینکه، امکان دارد در مواردی سرویس قابل اعتماد اساساً در دسترس نباشد. برای مثال، اترنت یک ارتباط قابل اعتماد نیست، و گاهی ممکنست داده‌ها در حین انتقال صدمه ببینند؛ این بر عهدهٔ پروتکلهای لایه‌های بالاتر است که این مشکل را حل کنند. دوم اینکه، تأخیر ذاتی سرویسهای مبتنی بر تصدیق دریافت در مواردی (مانند برنامه‌های چندرسانه‌ای) پذیرفتنی نیست. به این دلایل، وجود سرویسهای قابل اعتماد و غیرقابل اعتماد هر دو لازم است.

۱-۳-۴ عملکردهای پایه سرویس

هر سرویس با یک سری **عملکردهای پایه** (primitives) که در اختیار کاربر خود می‌گذارد، شناخته می‌شود. این عملکردهای پایه یا خود کاری را انجام می‌دهند، و یا انجام کاری را در طرف مقابل گزارش می‌کنند. اگر **پشته پروتکل** (protocol stack) جزئی از سیستم عامل باشد (که اغلب نیز چنین است)، عملکردهای پایه نیز معمولاً جزء فراخوانی‌های سیستم (system call) هستند. این فراخوانی‌ها باعث فعال شدن کُدی در هستهٔ سیستم عامل شده، و ارسال بسته‌های پیام انجام می‌شود.

عملکردهای پایهٔ هر سرویس به خصلت آن سرویس بستگی دارد. برای مثال، عملکردهای پایهٔ یک سرویس اتصال-گرا متفاوت از سرویسهای غیرمتصل است. در شکل ۱-۱۷ حداقل عملکردهای پایهٔ لازم برای پیاده‌سازی یک سرویس اتصال-گرای جریان بایت قابل‌اعتماد را در یک محیط مشتری-سرویس‌دهنده ملاحظه می‌کنید.

طرز استفاده از این عملکردهای پایه مانند زیر است. ابتدا، کامپیوتر سرویس‌دهنده LISTEN را اجرا می‌کند تا نشان دهد که آمادهٔ پذیرش ارتباطات ورودی است. عملکرد LISTEN معمولاً بصورت یک فراخوانی **مسدودشونده** (blocking) پیاده‌سازی می‌شود؛ بدین معنا که بعد از اجرای این عملکرد، پروسس سرویس‌دهنده تا زمان دریافت درخواست اتصال مسدود می‌شود.

سپس، مشتری عملکرد CONNECT را اجرا می‌کند تا به سرویس‌دهنده متصل شود. فراخوانی CONNECT معمولاً باید مشخص کند که مقصد اتصال کجاست، بهمن دلیل ممکنست پارامتری داشته باشد که آدرس سرویس‌دهنده را بدست می‌دهد. با این عمل، سیستم عامل مشتری پیامی را به همتای خود می‌فرستد، و درخواست اتصال می‌کند - این مرحله با شمارهٔ (1) در شکل ۱-۱۸ نشان داده شده است. پس از آن، پروسس مشتری تا زمان دریافت پاسخ به حالت **تعلیق** (suspend) درمی‌آید. وقتی این بسته به سرویس‌دهنده رسید، توسط سیستم عامل پردازش می‌شود، و وقتی می‌بیند یک درخواست اتصال است، بدنبال یک پروسس **شنونده** (listener) می‌گردد. اگر چنین پروسسی را پیدا کند، آنرا از حالت انسداد خارج کرده، و یک پیام **تصدیق دریافت** (acknowledgement) به مشتری پس می‌فرستند - مرحلهٔ (2) . دریافت این پیام توسط مشتری باعث می‌شود تا پروسس از حالت تعلیق در آید. در این لحظه پروسسهای سرویس‌دهنده و مشتری هر دو در حال اجرا هستند، و

عملکرد پایه	مفهوم
LISTEN	انتظاربرای دریافت اتصال
CONNECT	برقراری ارتباط با همتای منتظر
RECEIVE	انتظاربرای دریافت اتصال
SEND	ارسال پیام به همتا
DISCONNECT	پایان‌اتصال

شکل ۱-۱۷. پنج عملکرد پایهٔ لازم برای پیاده‌سازی یک سرویس اتصال-گرای ساده.



شکل ۱-۱۸. تبادل بسته‌ها در یک شبکهٔ اتصال-گرای مشتری-سرویس‌دهنده.

اتصال برقرار شده است. توجه به این نکته ضروریست که پیام تصدیق دریافت (2) توسط کُد پروتکل (که در **سطح هسته** - kernel level - اجرا می‌شود) ایجاد می‌شود، نه کُد عملکرد پایه (که یک پروسس **سطح کاربر** - user level - است). اگر هنگام دریافت درخواست اتصال توسط سرویس‌دهنده، هیچ پروسس شنونده‌ای وجود نداشته باشد، نتیجه نامشخص است. در برخی از سیستمها، این درخواست برای مدتی در **صف** (queue) می‌ماند، به امید اینکه شاید یک پروسس LISTEN اجرا شود.

این فرآیند بسیار شبیه تماس تلفنی مشتری با مدیر قسمت پشتیبانی مشتریان در یک شرکت است. نشستن مدیر قسمت پشتیبانی در کنار تلفن بنوعی اعلام آمادگی برای دریافت تقاضاها است (همان پروسس LISTEN). سپس یکی از مشتریان زنگ می‌زند (پروسس CONNECT)؛ و به محض اینکه مدیر پشتیبانی گوشی تلفن را برداشت، ارتباط برقرار می‌شود.

قدم بعدی را باید سرویس‌دهنده بردارد: اجرای عملکرد RECEIVE برای دریافت اولین درخواست. معمولاً سرویس‌دهنده این کار را بلافاصله بعد از برطرف شدن انسداد (و حتی قبل از اینکه پاسخ تصدیق دریافت آن به مشتری برسد) انجام می‌دهد. فراخوانی RECEIVE نیز جزء پروسسهای مسدودشونده است.

وقتی مشتری اعلام آمادگی سرویس‌دهنده را دریافت کرد، درخواست خود را در قالب یک عملکرد SEND به آن می‌فرستد - مرحلهٔ (3) .

رسیدن بستهٔ SEND به سرویس‌دهنده آنرا از حالت انسداد خارج کرده، و باعث می‌شود تا بتواند به درخواست مشتری رسیدگی کند. پس از آماده شدن پاسخ، سرویس‌دهنده آنرا با اجرای عملکرد SEND به مشتری پس می‌فرستد - مرحلهٔ (4) . رسیدن این بسته به مشتری باعث می‌شود تا از حالت انسداد خارج شده، و محتوای پاسخ را بررسی کند. اگر مشتری درخواست‌های بیشتری داشته باشد، آنها را در همین مرحله انجام می‌دهد؛ در غیر اینصورت، با اجرای عملکرد DISCONNECT اتصال را قطع می‌کند. فراخوانی DISCONNECT نیز معمولاً یک فراخوانی مسدودکننده است، و باعث تعلیق پروسس مشتری و اعلام پایان ارتباط به سرویس‌دهنده می‌شود - مرحلهٔ (5) . پروسس سرویس‌دهنده با دریافت پیام DISCONNECT از مشتری، عملکرد DISCONNECT را در سمت خود اجرا کرده، و (پس از اعلام به مشتری) ارتباط را قطع می‌کند. وقتی بستهٔ سرویس‌دهنده به مشتری رسید، مشتری نیز اتصال را رها می‌کند - مرحلهٔ (6) . این بود ماجرای سادهٔ یک ارتباط اتصال-گرا!

البته، زندگی همیشه این قدر شیرین نیست، و خیلی چیزها می‌تواند آنرا تلخ کند. برای مثال، فرض کنید عملکرد CONNECT قبل از LISTEN اجرا شود، یکی از بسته‌ها وسط راه گم شود، و اتفاقاتی از این قبیل. بعداً دربارهٔ این مسائل بیشتر صحبت خواهیم کرد، اما فعلاً به همین ارتباط سادهٔ شکل ۱-۱۸ دقت کنید.

با توجه به این نکته که برای کامل شدن سیکل این پروتکل به شش بسته نیاز داریم، شاید بپرسید که چرا بجای آن از یک ارتباط غیرمتصل (که فقط به دو بسته - یک درخواست و یک پاسخ - نیاز دارد) استفاده نکنیم. جواب اینست

که در یک دنیای بی عیب و نقص می‌توان چنین کرد؛ اما آیا اوضاع همیشه بر وفق مراد ماست؟ اگر فایل‌ها خیلی بزرگ باشند، خط ارتباطی پُر از خطا باشد، بسته‌ها مدام گم شوند، چکار باید کرد؟ اگر پای صدها و هزارها بسته در میان باشد، و فقط چند بستهٔ ناقابل گم شود، مشتری چگونه باید این موضوع را متوجه شود؟ مشتری از کجا بفهمد آخرین بسته‌ای که گرفته، واقعاً آخرین بسته بوده، یا خیلی ساده ارتباط قطع شده است؟ فرض کنید، مشتری فایل دومی را هم درخواست می‌کند، و بعد یکباره بسته‌ای می‌رسد که شمارهٔ ۱ دارد. آیا این اولین بستهٔ فایل دوم است، یا اولین بستهٔ فایل اول که تا حالا سرگردان بوده، و همین الان راه خود را پیدا کرده؟ خلاصه، در دنیای واقعی یک شبکهٔ غیرقابل‌اعتماد مبتنی بر پروتکل درخواست-پاسخ نمی‌تواند کافی باشد. در فصل ۳ پروتکلهایی را که برای رفع این مشکلات ابداع شده‌اند، به تفصیل بررسی خواهیم کرد. فعلاً همین قدر کافیست بدانید که، گاهی پروسسهایی که به روش جریان بایت با هم ارتباط برقرار می‌کنند، بهترین گزینه‌اند.

۱-۳-۵ رابطهٔ سرویس و پروتکل

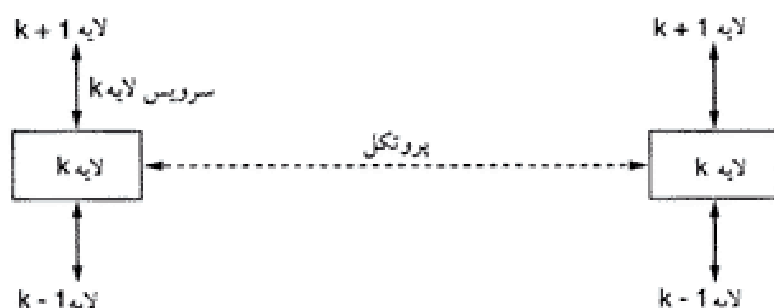
سرویس و پروتکل دو چیز متفاوتند، که اغلب افراد آنها را با هم اشتباه می‌گیرند. اما این تفاوت بقدری مهم است، که جا دارد باز هم بر آن تأکید کنیم. **سرویس** (service) عبارتست از مجموعه‌ای از عملکردهای پایه که یک لایه در اختیار لایهٔ بالاتر از خود قرار می‌دهد. سرویس فقط می‌گوید که یک لایه چه کارهایی می‌تواند برای کاربر خود انجام دهد، ولی هیچ چیز دربارهٔ چگونگی آن نمی‌گوید. سرویس در واقع به واسط بین دو لایه مربوط می‌شود، که در آن لایهٔ پائینتر ارائه‌دهندهٔ سرویس و لایهٔ بالاتر مصرف‌کنندهٔ سرویس است.

اما، **پروتکل** (protocol) عبارتست از مجموعه قواعد حاکم بر فرمت، مفهوم و نحوهٔ تبادل بسته‌ها و پیامها بین دو لایهٔ همتا. در واقع این پروتکل است که سرویسهای تعریف شده در هر لایه را پیاده‌سازی می‌کند. همتایان هر لایه می‌توانند پروتکل ارتباطی خود را عوض کنند، مشروط بر اینکه سرویسهایی که به کاربران خود می‌دهند، تغییری نکند. با این تعریف، سرویس و پروتکل کاملاً از هم مستقل هستند.

همانطور که در شکل ۱-۱۹ می‌بینید، سرویس به واسط دو لایه مربوط می‌شود، در حالیکه پروتکل پیامهای مبادله شده بین دو لایهٔ همتا (در دو کامپیوتر مختلف) را کنترل می‌کند. بسیار اهمیت دارد که این دو مفهوم را با هم مخلوط نکنید.

یک مثال مشابه از زبانهای برنامه‌نویسی می‌تواند در درک تمایز سرویس و پروتکل کمک کند. سرویس در واقع شبیه **نوع داده** (data type) یا **شیء** (object) در زبانهای شیء‌گرا است. با اینکه می‌دانیم یک شیء چه خواصی دارد، اما نمی‌دانیم آنها را چگونه پیاده‌سازی کرده است. این پیاده‌سازی همان پروتکل است که از چشم کاربر مخفی می‌ماند.

البته در بسیاری از پروتکلهای قدیمی این تمایز بروشنی وجود ندارد. در این پروتکلها هر تغییری در پروتکل بلافاصله به چشم کاربر خواهد آمد. اما امروزه طراحان شبکه سعی می‌کنند این تمایز را رعایت کنند.



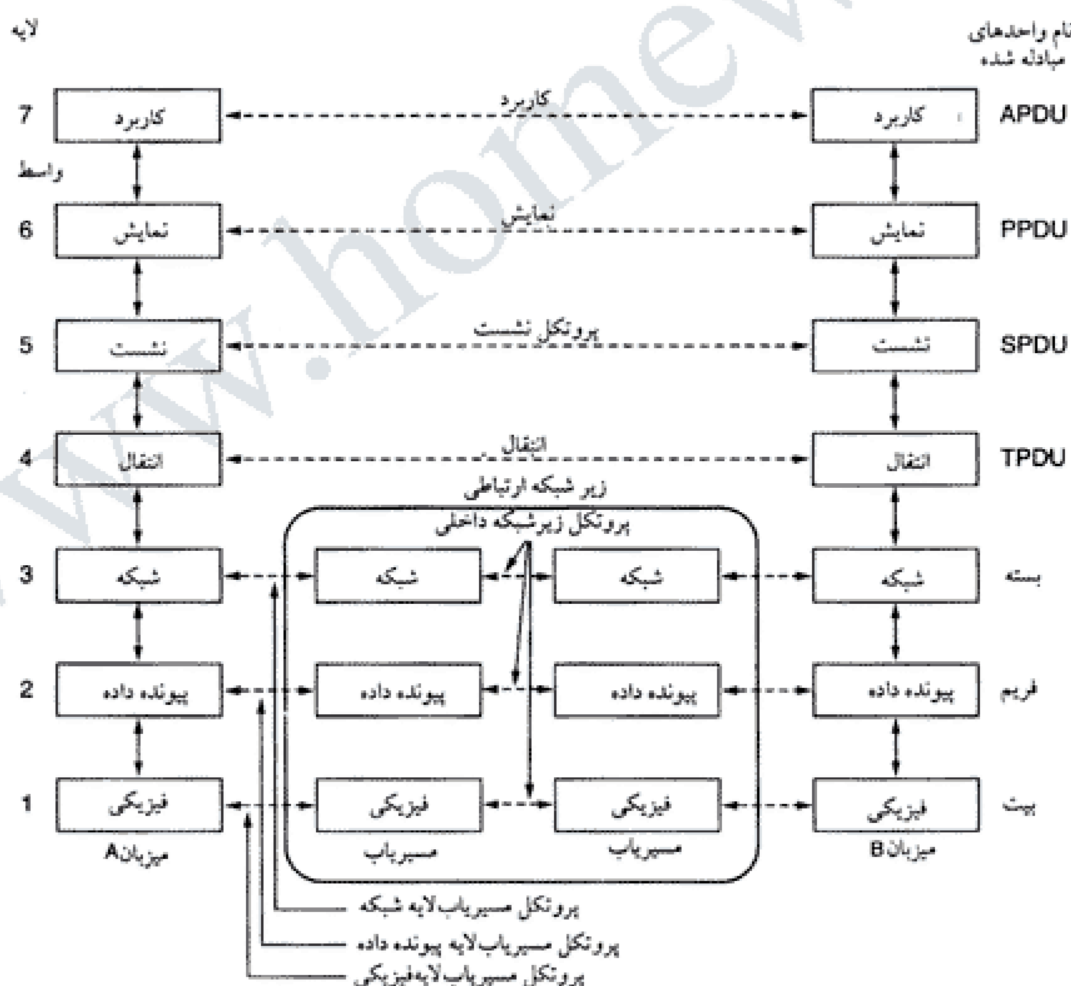
شکل ۱-۱۹. رابطهٔ بین سرویس و پروتکل.

۴ـ۱ مدلهای مرجع

حال که با شبکه‌های چند لایه بصورت تئوری آشنا شدید، وقت آنست که نگاهی به چند نمونه از این نوع شبکه‌ها بیندازیم. در دو قسمت آینده دو تا از مهمترین معماریهای شبکه، مدل مرجع OSI و مدل مرجع TCP/IP ، را بررسی خواهیم کرد. با اینکه پروتکلهای مدل OSI امروزه بندرت مورد استفادهٔ عملی دارند، اما این مدل همچنان معتبر، و مشخصات لایه‌های آن از اهمیت زیادی برخوردار است. وضعیت مدل TCP/IP بر عکس است: با اینکه خود این مدل کمتر مورد استفاده قرار می‌گیرد، ولی پروتکلهای آن کاربرد وسیعی دارند. به همین دلیل، مدلهای فوق را مفصلاً مورد بررسی قرار داده‌ایم.

۱ـ۴ـ۱ مدل مرجع OSI

در شکل ۱–۲۰ مدل OSI را (منهای لایهٔ رسانهٔ فیزیکی) ملاحظه می‌کنید. این مدل بر اساس نظرات پیشنهادی **سازمان بین‌المللی استانداردها** (International Standards Organization - ISO) – بعنوان اولین استاندارد بین‌المللی شبکه‌های چند لایه – توسعه داده شد (Day and Zimmermann, 1983) . این مدل در سال ۱۹۹۵ مورد تجدیدنظر قرار گرفت (Day, 1995) . این مدل که نام کامل آن **مدل مرجع ارتباطات سیستمهای باز** (ISO OSI - Open System Interconnection) است، با ارتباطات سیستمهای باز – سیستمهایی که قادر به ارتباط با سیستمهای دیگر هستند – سر و کار دارد؛ ولی ما آنرا بسادگی مدل OSI خواهیم نامید.



شکل ۱–۲۰. مدل مرجع OSI .

مدل OSI هفت لایه دارد. اما، چرا هفت لایه؟ برخی از مهمترین دلایل انتخاب هفت لایه عبارتند از:

۱. هر کجا به تجرید خاصی نیاز باشد، باید یک لایه ایجاد کرد.
۲. هر لایه باید وظیفهٔ کاملاً مشخصی را انجام دهد.
۳. وظیفهٔ هر لایه باید با در نظر گرفتن تعریف پروتکلهای استاندارد بین‌المللی انتخاب شود.
۴. مرزهای هر لایه باید طوری انتخاب شود، که کمترین انتقال اطلاعات از آنها لازم باشد.
۵. تعداد لایه‌ها باید آنقدر زیاد باشد، که نیازی به تعریف توابع مشابه در یک لایه نباشد؛ و باید آنقدر کم باشد که معماری شبکه بیش از حد بزرگ و پیچیده نشود.

در زیر لایه‌های مدل OSI را (از پائین به بالا) مورد بحث قرار داده‌ایم. توجه داشته باشید که مدل OSI خود یک معماری شبکه نیست، چون هیچ سرویس یا پروتکلی در آن تعریف نمی‌شود. این مدل فقط می‌گوید که هر لایه چه کاری باید انجام دهد. با اینکه سازمان استانداردهای بین‌المللی (ISO) پروتکلهای هر لایه را نیز تعریف کرده است، ولی آنها جزء مدل OSI نیستند، و جداگانه بصورت استانداردهای بین‌المللی منتشر می‌شوند.

لایهٔ فیزیکی

لایهٔ فیزیکی (physical layer) وظیفهٔ انتقال بیت‌های خام را از طریق کانال مخابراتی بر عهده دارد. مهمترین نکته در طراحی این لایه اینست که وقتی یک طرف یک بیت 1 می‌فرستد، طرف مقابل یک بیت 1 دریافت کند، نه یک بیت 0 . سؤالات اساسی در این لایه عبارتند از اینکه، برای 1 و 0 از چه ولتاژهایی استفاده کنیم، هر بیت باید چند نانوثانیه (یک میلیاردیم ثانیه) روی خط دوام بیاورد، آیا انتقال همزمان در هر دو جهت امکانپذیر باشد یا خیر، اتصال اولیه چگونه شروع شود و چگونه پایان یابد، **رابط شبکه** (network connector) چند پایه باید داشته باشد و وظیفهٔ هر پایه چیست. مسائل طراحی در این لایه عمدتاً از نوع مکانیکی، الکتریکی، تایمینگ (همزمانی)، و رسانهٔ فیزیکی انتقال (که زیر لایهٔ فیزیکی قرار دارد) هستند.

لایهٔ پیوند داده

مهمترین وظیفهٔ **لایهٔ پیوند داده** (data link layer) عبارتست از تبدیل خط فیزیکی پُر از خطا به یک خط ارتباطی عاری از خطا برای لایهٔ بالاتر، یعنی لایهٔ شبکه. لایهٔ پیوند داده این کار را با شکستن داده‌های ورودی به بسته‌های کوچک چند صد یا هزار بایتی (که **فریم داده** – data frame – نامیده می‌شوند)، و ارسال آنها انجام می‌دهد. وقتی گیرنده هر بسته را دریافت می‌کند، یک **فریم تصدیق دریافت** (acknowledgement frame) به فرستنده باز پس می‌فرستد، تا آنرا از دریافت صحیح بسته مطلع کند.

مسئلهٔ دیگری که در لایهٔ پیوند داده (و حتی لایه‌های بالاتر) باید حل شود اینست که، چگونه یک گیرندهٔ کُند را با یک فرستندهٔ سریع هماهنگ کند. برای این منظور باید مکانیزمی تعبیه شود تا فرستنده در هر لحظه از مقدار **بافر** (buffer – حافظهٔ موقتی) گیرنده مطلع باشد. در اغلب موارد این دو ویژگی – کنترل جریان اطلاعات و مقابله با خطا – در هم ادغام می‌شوند.

لایهٔ پیوند داده در شبکه‌های **پخشی** (broadcast) باید با مسئلهٔ دیگری نیز دست و پنجه نرم کند: کنترل دسترسی به یک کانال مشترک. برای این منظور از زیرلایه‌ای بنام **کنترل دسترسی رسانه** (medium access control) در لایهٔ پیوند داده استفاده می‌شود.

لایهٔ شبکه

لایهٔ شبکه (network layer) عملکرد زیرشبکه را کنترل می‌کند. یکی از مسائلی که باید در این لایه حل شود، نحوهٔ مسیریابی بسته‌ها از مبدأ به مقصد است. این مسیرها می‌توانند مسیرهای استاتیک باشند (مسیرهایی که بطور

ثابت و بندرت متغیر در شبکه تعبیه شده‌اند)، یا مسیرهای نیمه‌استاتیک (مسیرهایی که در ابتدای هر نشست تعیین و مشخص می‌شوند)، و یا مسیرهای دینامیک (مسیرهایی که در هر لحظه و برای هر بسته از نو - و با توجه به بار شبکه - جستجو و مشخص می‌شوند).

اگر تعداد بسته‌های در حال حرکت در یک زیرشبکه بیش از حد باشد، آنها راه یکدیگر را بند آورده و وضعیتی را بوجود می‌آورند که به آن **گلوگاه** (bottleneck) یا **ازدحام** (congestion) گفته می‌شود. کنترل این وضعیت نیز بر عهدهٔ لایهٔ شبکه است. به بیان کلی، کیفیت سرویس (تأخیر انتشار بسته‌ها، زمان انتقال آنها، و حالتهای گذرا در شبکه) همگی جزء مسئولیتهای لایهٔ شبکه است.

اگر یک بسته برای رسیدن به مقصد خود باید از یک شبکه خارج و وارد شبکهٔ دیگری شود، مسائل جدیدی بروز خواهد کرد. اول اینکه، امکان دارد آدرس‌دهی در این شبکه‌ها متفاوت باشد. دوم اینکه، ممکنست شبکهٔ دوم این بسته را بکلی نپذیرد، چون مثلاً اندازهٔ آن بیش از حد بزرگ است؛ و یا اینکه پروتکلها با هم فرق داشته باشند، و مسائلی از این قبیل. حل همهٔ این مسائل (و بهم پیوستن شبکه‌های ناهمگن) نیز از وظایف لایهٔ شبکه است.

در شبکه‌های پخشی مسیریابی بسته‌ها ساده است، بهمین دلیل لایهٔ شبکه یا بسیار کوچک است یا اصلاً نیست

لایهٔ انتقال

اصلی‌ترین وظیفهٔ **لایهٔ انتقال** (transport layer) گرفتن داده‌ها از لایهٔ بالاتر، تقسیم آن به قطعات کوچکتر (در صورت نیاز)، ارسال آن به لایهٔ شبکه، و حصول اطمینان از دریافت صحیح آنها در طرف مقابل است. علاوه بر آن، همهٔ این کارها باید بگونه‌ای مؤثر و طوری انجام شود که لایه‌های بالاتر را از تغییرات اجتناب‌ناپذیر در سخت‌افزار ایزوله کند.

این لایه همچنین تعیین می‌کند که چه سرویسهایی باید در اختیار لایهٔ نشست (و از آنجا، در اختیار کاربران شبکه) قرار گیرد. متداولترین نوع انتقال کانالهای نقطه-به-نقطهٔ عاری از خطاست، که در آن بایتها بهمان ترتیبی که فرستاده شده‌اند، در طرف مقابل دریافت می‌شوند. با این حال انواع دیگری از سرویسهای انتقال وجود دارد، که از میان آنها می‌توان به انتقال پیام بدون اطمینان از دریافت منظم آنها، و ارسال همزمان پیامهای پخشی به چندین نقطه اشاره کرد. نوع این سرویسها در لحظهٔ برقراری اتصال مشخص می‌شود. (البته باید تصریح کنیم که کانال عاری از خطا در دنیای واقعی وجود خارجی ندارد، و آنچه از این اصطلاح برداشت می‌شود پائین بودن نرخ خطاست، بگونه‌ای که بتوان در عمل آنرا نادیده گرفت.)

لایهٔ انتقال یک لایهٔ نقطه-به-نقطهٔ واقعی است، که در آن کامپیوتر فرستنده (مبدأ) مستقیماً با کامپیوتر گیرنده (مقصد) ارتباط دارد. در لایه‌های پائینتر ارتباط ماشین مبدأ معمولاً با ماشینهای همسایه (و نه ماشین مقصد) است. تفاوت لایه‌های ۱ تا ۳ (که بصورت زنجیره‌ای هستند) با لایه‌های ۴ تا ۷ (که نقطه-به-نقطه هستند) در شکل ۱-۲۰ نشان داده شده است.

لایهٔ نشست

لایهٔ نشست (session layer) اجازه می‌دهد تا بین کاربران در ماشینهای مختلف نشست برقرار شود. نشست سرویسهای مختلفی ارائه می‌کند، از جمله: **کنترل دیالوگ** (dialog control - کنترل اینکه نوبت چه کسی است)، **مدیریت نشانه** (token management - جلوگیری از تداخل اعمال مهم)، و **سنکرون کردن** (synchronization - کنترل عملیات انتقال طویل‌المدت، و از سرگیری آن از نقطهٔ قطع شده در صورت بروز اختلال).

لایهٔ نمایش

بر خلاف لایه‌های پائینتر، که عمدتاً با بیت‌ها سروکار دارند، **لایهٔ نمایش** (presentation layer) توجه خود را

روی ساختار پیامها و مفهوم آنها متمرکز می‌کند. برای اینکه کامپیوترهایی با ساختارهای متفاوت داده بتوانند با هم ارتباط برقرار کنند، ساختار پیامهای مبادله شده بایستی کاملاً مشخص و استاندارد باشد. وظیفهٔ لایهٔ نمایش مدیریت این ساختارها در سطح بالاست.

لایهٔ کاربرد

بسیاری از پروتکلهای موردنیاز کاربران در **لایهٔ کاربرد** (application layer) قرار دارد، که از معروفترین آنها می‌توان به **پروتکل HTTP** (HyperText Transfer Protocol) - پروتکل اصلی وب - اشاره کرد. وقتی مرورگر وب می‌خواهد صفحه‌ای را بار کند، نام آن صفحه را با استفاده از پروتکل HTTP به سرویس‌دهندهٔ وب می‌فرستد؛ سرویس‌دهندهٔ وب نیز با همین پروتکل صفحه را به مرورگر برمی‌گرداند. **پروتکل انتقال فایل** (FTP) ، **پروتکل انتقال خبر** (NNTP) ، و **پروتکلهای پُست الکترونیک** (SMTP و POP) نیز جزء پروتکلهای کاربردی هستند.

۲-٤-۱ مدل مرجع TCP/IP

اکنون اجازه دهید به مدل مرجع بکار رفته در پدربزرگ شبکه‌های کامپیوتری، **آرپانت** (ARPANET) ، و خلف آن اینترنت، بپردازیم. آرپانت یک شبکهٔ تحقیقاتی بود که توسط DoD (وزارت دفاع ایالات متحدهٔ آمریکا) پایه‌ریزی شد. بتدریج صدها دانشگاه و مرکز دولتی بوسیلهٔ خطوط اجاره‌ای تلفن (leased line) به این شبکه ملحق شدند. با پیشرفت مخابرات رادیویی و ماهواره، مشکلاتی در پروتکلهای ارتباطی آرپانت بوجود آمد، که انتخاب یک معماری مرجع جدید را الزامی می‌کرد. یکی از اولین هدفهای آرپانت ارتباط یکپارچهٔ شبکه‌های مختلف بود، که بالاخره (بعد از بررسی چندین پروتکل) توسط **مدل مرجع TCP/IP** محقق شد. این مدل برای اولین بار توسط (Cerf and Kahn, 1974) تعریف شد، که در سال ۱۹۸۵ مورد تجدیدنظر قرار گرفت (Leiner et al., 1985) . فلسفهٔ طراحی مدل مرجع TCP/IP در (Clark, 1988) مورد بحث قرار گرفته است.

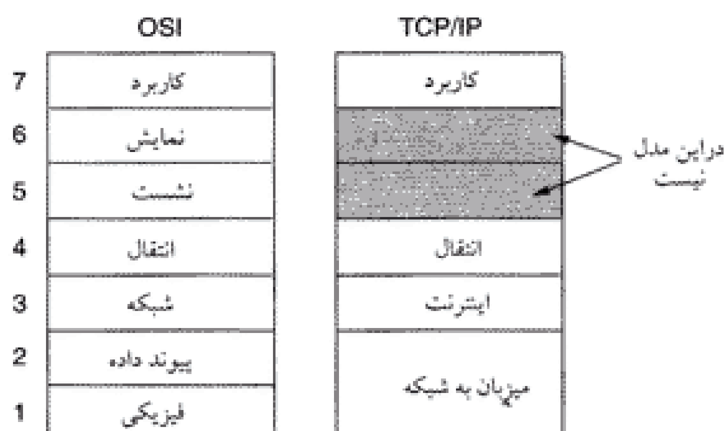
دغدغهٔ همیشگی وزارت دفاع این بوده که بخشی از شبکه و وسایل با ارزش آن در یک لحظه (احتمالاً در یک حملهٔ اتمی) نیست و نابود شود، و بهمین دلیل همواره تأکید داشته که این شبکه باید بگونه‌ای طراحی شود که حتی در صورت از بین رفتن بخشی از زیرشبکه‌های آن، بتواند بدون وقفه به کار خود ادامه دهد. بعبارت دیگر، هدف این است که دو کامپیوتر مادامیکه که کار می‌کنند، باید بتوانند با هم ارتباط داشته باشند (حتی اگر تعدادی از ماشینهای واسط بین آنها از مدار خارج شوند). علاوه بر آن، این مدل باید بتواند از عهدهٔ طیف وسیعی از کاربردهای متنوع (از انتقال فایل گرفته، تا مکالمهٔ زمان واقعی) برآید.

لایهٔ اینترنت

تمام این الزامات باعث شد تا در نهایت یک شبکهٔ **سوئیچینگ بسته** (packet-switching) مبتنی بر یک لایهٔ ارتباطات **غیرمتصل** (connectionless) انتخاب شود. این لایه، که **لایهٔ اینترنت** (internet layer) نام دارد، سنگ بنای معماری TCP/IP است. وظیفهٔ این لایه اینست که به ماشینها اجازه دهد بسته‌های خود را روی شبکه و به سمت مقصد بفرستند. این لایه رسیدن پیامها را با همان ترتیبی که فرستاده شده‌اند، تضمین نمی‌کند؛ وظیفهٔ مرتب کردن پیامها (در صورت نیاز) بر عهدهٔ لایه‌های بالاتر است. (دقت کنید که در اینجا «اینترنت» یک کلمهٔ عام است.)

لایهٔ اینترنت تا حد زیادی شبیه سیستم پُست است. اگر چند نامه را به مقصد کشوری دیگر در صندوق پُست بیندازید، با کمی شانس همهٔ آنها به دست گیرنده خواهند رسید. البته احتمال دارد که هر یک از این نامه از مسیر متفاوتی به مقصد رسیده باشد، که این موضوع از دید کاربران پنهان است (و علاقه‌ای هم به دانستن آن ندارند).

فرمت بسته‌های پیام و پروتکل آنها در لایهٔ اینترنت تعریف می‌شود، که **IP** (Internet Protocol) نام دارد.



شکل ۱-۲۱. مدل مرجع TCP/IP .

وظیفهٔ لایهٔ اینترنت اینست که بسته‌های IP را به مقصد برساند. مسیریابی بسته‌ها (و جلوگیری از ازدحام در مسیرهای شلوغ) برعهدهٔ این لایه است، که از این نظر می‌توان آنرا معادل لایهٔ شبکه در مدل OSI دانست (شکل ۱-۲۱ را ببینید).

لایهٔ انتقال

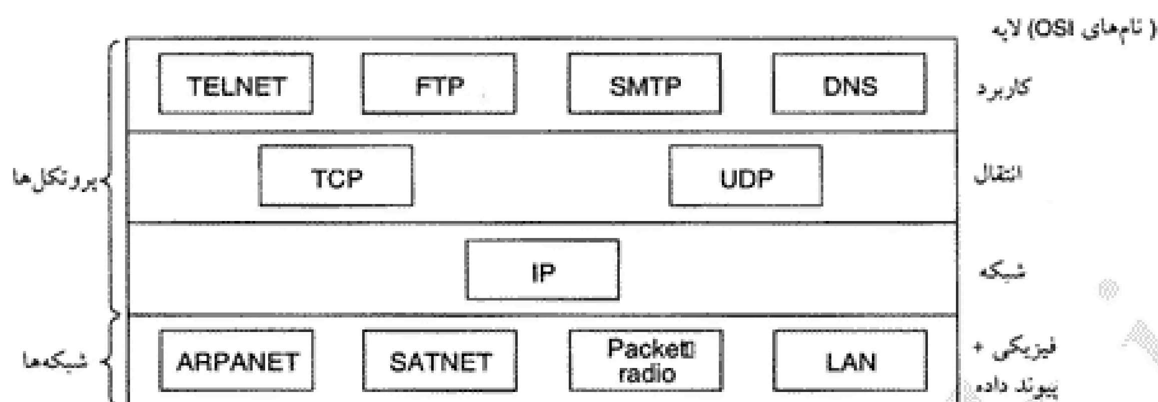
لایهٔ بالای لایهٔ اینترنت در مدل TCP/IP ، **لایهٔ انتقال** (transport layer) نام دارد. این لایه شبیه لایهٔ انتقال در مدل OSI است، و اجازه می‌دهد تا عناصر همتا در کامپیوترهای مبدأ و مقصد با هم مکالمه انجام دهند. در این لایه دو پروتکل انتقال نقطه-به-نقطه تعریف شده است. پروتکل اول، که TCP (Transmission Control Protocol) نام دارد، یک پروتکل اتصال-گرای قابل اعتماد است که اجازه می‌دهد تا جریانی از بایتها بدون خطا از یک کامپیوتر در اینترنت به کامپیوتر دیگر فرستاده شود. این پروتکل جریان بایت‌ها را بصورت بسته بسته در آورده، و به لایهٔ اینترنت تحویل می‌دهد. در ماشین مقصد عکس این عمل انجام می‌شود: بسته‌ها به هم چسبانده شده، و بصورت جریانی از بایتها به لایهٔ بالاتر فرستاده می‌شود. در پروتکل TCP **کنترل جریان داده‌ها** (flow control) نیز وجود دارد، بدین معنا که فرستنده داده‌ها را سریعتر از آنچه گیرنده توان دریافت آنرا دارد، ارسال نخواهد کرد.

پروتکل دوم این لایه، که **UDP** (User Datagram Protocol) نام دارد، یک پروتکل غیرمتصل غیر قابل اعتماد است، که در مواردی که نیازی به سخت‌گیریهای TCP نیست از آن استفاده می‌شود. این پروتکل بیشتر در مواردی که سرعت اهمیت بیشتری دارد تا دقت (مانند انتقال صوت و تصویر)، یا در جاهائیکه فرآیند درخواست-پاسخ فقط یک بار انجام می‌شود، بکار می‌رود. در شکل ۱-۲۲ رابطهٔ پروتکلهای IP ، TCP و UDP را مشاهده می‌کنید. پروتکل IP اکنون در شبکه‌های بسیاری پیاده‌سازی شده است.

لایهٔ کاربرد

مدل TCP/IP لایه‌های نشست یا نمایش ندارد، یعنی در واقع معتقد است که نیازی به آنها نیست. تجربهٔ مدل OSI نیز نشان می‌دهد که این نظر درست است، و این دو لایه بندرت کاربردی پیدا کرده‌اند.

در بالای لایهٔ انتقال **لایهٔ کاربرد** (application layer) قرار می‌گیرد، که تمام پروتکلهای سطح-بالا در آن قرار دارند. پروتکلهای **ترمینال مجازی** (TELNET) ، **انتقال فایل** (FTP) و **پُست الکترونیک** (SMTP) از پروتکلهایی هستند که از سالها قبل در این لایه پیاده‌سازی شده‌اند (شکل ۱-۲۲). پروتکل ترمینال مجازی اجازه



شکل ۱-۲۲. پروتکل‌ها و شبکه‌ها در مدل TCP/IP .

می‌دهد تا کاربر وارد کامپیوترهای راه دور شده، و با آنها مانند یک کامپیوتر محلی کار کند. پروتکل انتقال فایل نیز ابزاریست مؤثر برای انتقال اطلاعات از یک ماشین به ماشین دیگر. پُست الکترونیک در ابتدا چیزی بیش از یک انتقال فایل ساده نبود، ولی بعدها یک پروتکل خاص بنام SMTP برای آن توسعه داده شد. اکنون پروتکلهای معروف دیگری نیز در این لایه وجود دارند، که برخی از آنها عبارتند از: پروتکل **نام ناحیه** (DNS) برای ترجمهٔ نام کامپیوترها به آدرس شبکه، پروتکل **انتقال خبر** (NNTP) برای خواندن مقالات **یوزنت** (USENET) ، پروتکل **انتقال صفحات ابَرمتن** (HTTP) برای خواندن صفحات وب، و دهها پروتکل دیگر.

لایهٔ میزبان - به - شبکه

در زیر لایهٔ اینترنت یک شکاف بزرگ دیده می‌شود؛ در واقع مدل TCP/IP دربارهٔ این قسمت تا حد زیادی سکوت کرده است، و فقط انتظار دارد که میزبان بنحوی به شبکه وصل شده، و بتواند بسته‌های IP را ارسال کند. پروتکل انجام این کار در مدل TCP/IP تعریف نمی‌شود، و در موارد مختلف متفاوت است (حتی کتابها و مقالاتی که دربارهٔ TCP/IP نوشته شده‌اند، بندرت در این باره صحبت می‌کنند).

۱-٤-۳ مقایسهٔ مدلهای OSI و TCP/IP

مدلهای TCP/IP و OSI نقاط مشترک زیادی دارند. هر دوی آنها مبتنی بر مجموعه‌ای از پروتکلهای مستقل هستند، و عملکرد لایه‌ها نیز تا حد زیادی شبیه یکدیگر است. برای مثال، در هر دو مدل لایه‌های بالای لایهٔ انتقال (و از جمله خود آن) بصورت نقطه-به-نقطه عمل می‌کنند، مستقل از شبکه هستند، و سرویسهای خود را (به شکلی کاربرد-گرا) در اختیار لایه‌های بالاتر می‌گذارند.

علیرغم شباهتهای اساسی، این دو مدل تفاوتهای بسیاری نیز با هم دارند، که در این قسمت به آنها خواهیم پرداخت. شایان ذکر است که ما در اینجا مدلها را با هم مقایسه می‌کنیم، نه مجموعه پروتکلهای آنها را (در این باره نیز بعداً صحبت خواهیم کرد). برای یک مقایسهٔ کامل و جامع بین TCP/IP و OSI به کتاب (Piscitello and Chapin, 1993) مراجعه کنید.

در مدل OSI سه مفهوم محوری وجود دارد:

۱. سرویس (service)
۲. واسط (interface)
۳. پروتکل (protocol)

شاید بزرگترین دستاورد مدل OSI روشن ساختن مفاهیم فوق (و تفکیک آنها) باشد. هر لایه سرویسهایی در اختیار لایه‌های بالاتر از خود قرار می‌دهد. تعریف این سرویسها فقط می‌گوید که یک لایه چه کاری انجام می‌دهد، و هیچ حرفی دربارهٔ نحوهٔ انجام آنها و چگونگی استفاده از سرویسها نمی‌زند.

تعریف چگونگی دسترسی به سرویسهای یک لایه بر عهدهٔ واسط است. واسط پارامترهای ورودی لازم، و نتیجه‌ای را که باید منتظر آن باشید، تعریف می‌کند. حتی واسط هم نمی‌گوید که یک لایه چگونه کار خود را انجام می‌دهد.

و بالاخره، کاری که لایه انجام می‌دهد را پروتکل‌های آن لایه تعریف می‌کنند. یک لایه مادامیکه کارش را بدرستی انجام دهد، می‌تواند از هر پروتکلی استفاده کند. تغییر پروتکلهای یک لایه هیچ تأثیری روی ارتباط آن با لایه‌های بالاتر نخواهد گذاشت.

ایده‌های فوق بسیار شبیه مفاهیم مدرن برنامه‌نویسی شیء‌گرا هستند. هر شیء، مانند یک لایه، متدها (عملکردها)یی دارد که اشیاء دیگر از آنها استفاده می‌کنند. نحوهٔ استفاده از این متدها در واقع همان سرویسهائیست که این شیء در اختیار دیگران می‌گذارد. ورودیها و خروجیهای شیء واسط آن با دنیای خارج هستند. کُد اجرایی شیء نیز شبیه همان پروتکل است، که نحوهٔ عملکرد آن از دید دیگران مخفی است.

در مدل اولیهٔ TCP/IP تمایز بین سرویسها، واسطها و پروتکلها واضح و مشخص نبود، اگرچه افرادی (با توجه به تجربهٔ موفق OSI) سعی کرده بودند آنرا هر چه بیشتر شبیه OSI کنند. برای مثال، لایهٔ اینترنت فقط دو سرویس واقعی بنامهای SEND IP PACKET و RECEIVE IP PACKET داشت. با توجه به این وضع، پروتکلهای OSI بهتر از TCP/IP مخفی شده‌اند، و امکان تغییر آنها براحتی وجود دارد، چیزی که هدف غایی طراحی لایه‌ای محسوب می‌شود.

مدل OSI قبل از اختراع پروتکلهای آن طراحی و ابداع شد. این بدان معناست که مدل OSI وابستگی و تمایل خاصی به هیچ مجموعه پروتکلی ندارد، چیزی که در سایر مدلها بسیار دیده می‌شود. البته این وضعیت یک نقطه ضعف نیز دارد، و آن اینست که طراحان تجربهٔ چندانی در زمینهٔ موضوع کار ندارند، و واقعاً نمی‌دانند کدام عملکرد را باید در کدام لایه قرار دهند. برای مثال، لایهٔ پیوند داده در ابتدا فقط برای شبکه‌های نقطه-به-نقطه طراحی شده بود، و وقتی شبکه‌های پخشی وارد بازار شد، مجبور شدند یک زیرلایه به آن اضافه کنند.

وقتی افراد شروع به طراحی شبکه با استفاده از مدل OSI و پروتکلهای موجود کردند، بزودی دریافتند که این شبکه‌ها با سرویسهای موردنیاز انطباق ندارند(!)، بنابراین مجبور شدند زیرلایه‌های زیادی به آن وصله‌پینه کنند. بالاخره، کمیتهٔ استاندارد مقرر کرد که هر کشور برای خود یک شبکهٔ منطبق با مدل OSI (تحت نظارت دولت) داشته باشد - شبکه‌ای که به هیچ عنوان آینده (اینترنت) در آن دیده نشده بود. خلاصه، کارها آنطوری که انتظار داشتند از آب در نیامد.

در مورد TCP/IP وضع بر عکس بود: اول پروتکلها اختراع و توسعه داده شدند، و سپس مدلی برای توصیف آنها ساخته شد. هیچ مشکلی در زمینهٔ انطباق پروتکلها با مدل وجود نداشت؛ همه چیز جفت و جور بود. تنها مشکل این بود که این مدل با هیچ مجموعه پروتکل دیگری جور در نمی‌آمد. این بدان معنا بود که مدل TCP/IP بدرد توصیف شبکه‌های غیر TCP/IP نمی‌خورد.

جدای از مسائل فلسفی قضیه، تفاوت دیگر در تعداد لایه‌های این دو مدل است: مدل OSI هفت لایه دارد و مدل TCP/IP چهار لایه. لایه‌های شبکه، انتقال و کاربرد در هر دو مشترکند، ولی لایه‌های دیگر فرق دارند.

تفاوت دیگر در زمینهٔ ارتباطات اتصال-گرا و غیرمتصل است. مدل OSI از هر دو نوع ارتباط اتصال-گرا و غیرمتصل در لایهٔ شبکه پشتیبانی می‌کند، ولی در لایهٔ انتقال فقط سرویس اتصال-گرا دارد (چون این سرویس در

معرض دید کاربران است). مدل TCP/IP در لایهٔ شبکه فقط سرویس غیرمتصل دارد، ولی در لایهٔ انتقال از هر دو نوع ارتباط پشتیبانی می‌کند، و دست کاربر را برای انتخاب باز می‌گذارد (که بویژه برای پروتکلهای سادهٔ درخواست-پاسخ بسیار مهم است).

۱-۴-۴ نگاهی انتقادی به مدل OSI و پروتکلهای آن

مدل‌های OSI و TCP/IP (و پروتکلهایشان) هیچکدام کامل نیستند، و جا دارد برخی از نقاط ضعف آنها را برشماریم. در این قسمت و قسمت آینده، برخی از نقاط ضعف مهم مدلهای OSI و TCP/IP را بررسی خواهیم کرد. با مدل OSI شروع می‌کنیم.

در زمان چاپ ویرایش دوم این کتاب (سال ۱۹۸۹)، بسیاری از متخصصان برجستهٔ شبکه بر این باور بودند که آینده دربست متعلق به مدل OSI و پروتکلهای آن است، و هیچ چیز نمی‌تواند در مقابل پیشرفت آن مقاومت کند. اما این اتفاق نیفتاد. چرا؟ نگاهی به گذشته درسهای بسیاری برای چشمان عبرت‌بین دارد، که می‌توان آنها را چنین خلاصه کرد:

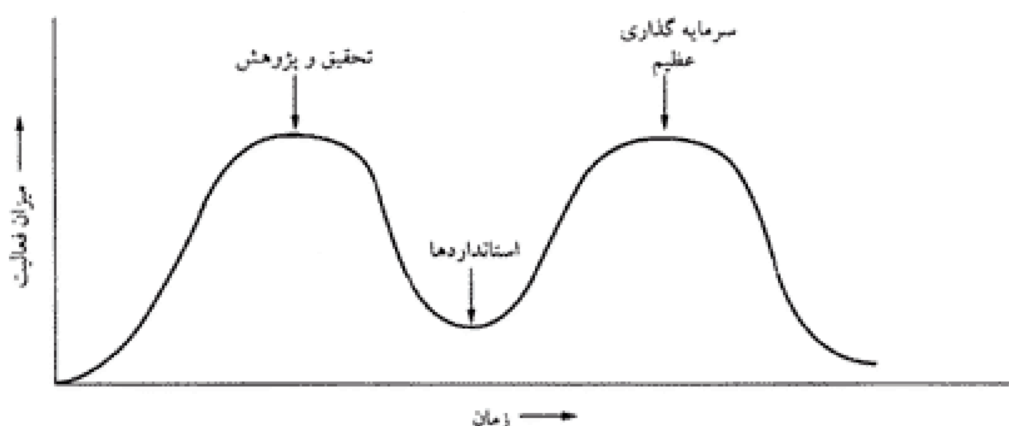
۱. زمان نامناسب
۲. تکنولوژی نامناسب
۳. پیاده‌سازی نامناسب
۴. سیاست‌های نامناسب

زمان نامناسب

اولین عامل شکست مدل OSI زمان نامناسب بود. زمانی که یک استاندارد وضع می‌شود، اهمیت حیاتی در موفقیت یا عدم موفقیت آن دارد. دیوید کلارک از دانشگاه M.I.T فرضیه‌ای در زمینهٔ استانداردها دارد که به ملاقات فیل‌ها معروف است، و در شکل ۱-۲۳ آنرا مشاهده می‌کنید.

این شکل میزان فعالیتهای حول یک موضوع جدید را نشان می‌دهد. وقتی موضوعی برای اولین بار کشف می‌شود، گرداگرد آنرا سیلی از فعالیتهای تحقیقی (به شکل بحث، مقاله و سخنرانی) فرا می‌گیرد. بعد از مدتی این موج فروکش می‌کند، و بعد از اینکه صنعت به آن موضوع علاقمند شد، موج سرمایه‌گذاری‌ها از پی می‌آید.

بسیار مهم است که در نقطهٔ تلاقی این دو فیل (موج تحقیق و موج سرمایه‌گذاری) استانداردها بطور کامل وضع شوند. اگر استاندارد زودتر از موعد (قبل از پایان تحقیقات) نوشته شود، خطر آن هست که موضوع بدرستی



شکل ۱-۲۳. فرضیهٔ ملاقات فیل‌ها

درک نشده باشد، و استاندارد ضعیف از آب در آید. اگر استاندارد دیرتر از موعد (بعد از شروع موج‌سرمایه‌گذاری) نوشته شود، شرکتهای بسیاری قبلاً - در مسیرهای مختلف - در آن سرمایه‌گذاری کرده‌اند، و این خطر هست که استاندارد آنها را نادیده بگیرد. اگر فاصلهٔ این دو فیل خیلی کم باشد (همه عجله داشته باشند که زودتر کار را شروع کنند)، خطر آن هست که استانداردنویسان بین آنها له شوند.

اکنون معلوم شده است که پروتکلهای استاندارد OSI بین فیلها له شدند. وقتی پروتکلهای OSI پا به عرصهٔ وجود گذاشتند، پروتکلهای رقیب (TCP/IP) مدتها بود که در دانشگاهها و مراکز تحقیقاتی پذیرفته شده بودند. با اینکه هنوز موج سرمایه‌گذاری صنعتی در TCP/IP شروع نشده بود، اما بازار آکادمیک آنقدر بزرگ بود که شرکتهای بسیاری را تشویق به تولید محصولات TCP/IP کند. و وقتی OSI بالاخره از راه رسید، کسی نبود که داوطلبانه از آن پشتیبانی کند. همه منتظر بودند دیگری قدم اول را بردارد؛ قدمی که هرگز برداشته نشد، و OSI در نطفه خفه شد.

تکنولوژی نامناسب

دلیل دیگری که OSI هرگز پا نگرفت آن بود که، این مدل و پروتکلهای آن هر دو ناقص و معیوب بودند. انتخاب هفت لایه برای این مدل بیشتر یک انتخاب سیاسی بود تا فنی، و در حالیکه دو لایهٔ آن (نشست و نمایش) تقریباً خالی بودند، در لایه‌های دیگر (لینک داده و شبکه) جای نفس کشیدن نبود.

مدل OSI (و سرویسها و پروتکلهای آن) بطرزی باورنکردنی پیچیده است. اگر کاغذهای چاپی این استاندارد را روی هم بچینید، ارتفاع آن از نیم متر هم بیشتر خواهد شد! پیاده‌سازی پروتکلهای OSI بسیار دشوار، و عملکرد آنها ناقص است. در این رابطه، نقل جملهٔ جالبی از پاول موکاپتریس (Rose, 1993) خالی از لطف نیست:

سؤال: از ترکیب یک گانگستر با یک استاندارد بین‌المللی چه چیزی بدست می‌آید؟

جواب: کسی پیشنهادی به شما می‌کند که از آن سر در نمی‌آورید.

مشکل دیگر مدل OSI ، علاوه بر غیر قابل فهم بودن آن، اینست که برخی از عملکردهای آن (مانند آدرس‌دهی، کنترل جریان داده‌ها، و کنترل خطا) در تمام لایه‌ها تکرار می‌شود. برای مثال، سالتزر و همکارانش (1984) نشان دادند که کنترل خطا باید در بالاترین لایه انجام شود تا بیشترین تأثیر را داشته باشد، بنابراین تکرار آن در لایه‌های پائینتر نه تنها غیرضروری است، بلکه باعث افت کارایی هم خواهد شد.

پیاده‌سازی نامناسب

با توجه به پیچیدگی بیش از حد مدل OSI و پروتکلهای آن، جای تعجب نبود که اولین پیاده‌سازیهای آن حجیم، سنگین و کُند باشد. آنهایی که با این مدل کار کرده بودند، بزودی پشیمان شدند، و طولی نکشید که کلمهٔ OSI مترادف شد با «کیفیت بد». بعدها محصولات بهتری به بازار آمد، اما آوازهٔ منفی OSI فراموش نشد.

از طرف دیگر، اولین پیاده‌سازی TCP/IP (که بخشی از سیستم عامل **یونیکس برکلی** بود) بسیار خوب از کار در آمد (و لازم به گفتن نیست که مجانی هم بود). افراد بسیاری بسرعت شروع به استفاده از آن کردند، هواخواه آن شدند، آنرا توسعه دادند، و این باعث شد که باز هم به خیل طرفداران آن اضافه شود. در اینجا، برخلاف OSI ، مارپیچ رو به بالا می‌رفت، نه پائین.

سیاست‌های نامناسب

بدلیل اولین پیاده‌سازی TCP/IP ، بسیاری از افراد (بویژه در محیطهای دانشگاهی) تصور می‌کردند که TCP/IP جزئی از یونیکس است، و یونیکس هم در آن دوران محبوبیتی فوق‌العاده داشت.

از سوی دیگر، این عقیده رواج داشت که OSI یک مخلوق دولتی (مخصوصاً دولتهای اروپایی و آمریکایی) است. البته این عقیده فقط تا حدی درست بود، اما همین تصور هم که عده‌ای دیوانسالار دولتی بخواهند یک

استاندارد فنی را بزور جا بیندازند، باعث شد تا برنامه‌نویسان و طراحان شبکه تمایلی به همکاری از خود نشان ندهند. زبانهای برنامه‌نویسی PL/1 (که در دهۀ ۱۹۶۰ از سوی IBM بعنوان زبان آینده توسعه داده شد) و Ada (که وزارت دفاع آمریکا حامی آن بود) بهمین دلیل دچار سرنوشتی مشابه شدند.

۱-۴-۵ نگاهی انتقادی به مدل TCP/IP

مدل TCP/IP و پروتکلهای آن نیز مشکلات خاص خود را دارند. اول اینکه، در این مدل مفاهیم سرویس، واسط و پروتکل بروشنی از یکدیگر تفکیک نشده‌اند؛ کاری که در مدل OSI بخوبی انجام شده است. به همین دلیل نمی‌توان از TCP/IP بعنوان ابزاری برای طراحی و توسعۀ شبکه‌های جدید استفاده کرد.

دوم اینکه، مدل TCP/IP به هیچ عنوان یک مدل کلی نیست، و نمی‌توان از آن برای توصیف شبکه‌های غیر TCP/IP استفاده کرد. برای مثال، توصیف **بلوتوث** با مدل TCP/IP بکلی غیرممکن است.

سوم اینکه، با در نظر گرفتن مفاهیم شبکه‌های چند لایه، **لایۀ میزبان-به-شبکه** اساساً یک لایۀ واقعی نیست، بلکه فقط یک واسط (بین لایه‌های شبکه و لینک داده) است. در واقع، این یکی از مهمترین جاهائیست که مدل TCP/IP مفاهیم واسط و لایه را با هم قاطی کرده است.

چهارم اینکه، در مدل TCP/IP هیچ تمایزی بین لایه‌های فیزیکی و لینک داده نیست (و حتی حرفی از آنها بمیان نیامده است). اینها دو لایۀ کاملاً متفاوت هستند - لایۀ فیزیکی با مشخصات کابل و فیبر نوری و کانالهای مخابراتی سروکار دارد، در حالیکه وظیفۀ لایۀ پیوند داده شکستن داده‌ها به قطعات کوچکتر و اطمینان از تحویل صحیح آنها به مقصد است. در یک مدل کامل این دو لایه باید از هم جدا باشند؛ کاری که در مدل TCP/IP انجام نشده است.

و بالاخره اینکه، اگر چه پروتکلهای TCP و IP بسیار خوب طراحی و پیاده‌سازی شده‌اند، بسیاری از دیگر پروتکلهای این مدل چنین نیستند، و اغلب توسط دانشجویان کنجکاو (و در ساعات بیکاری) نوشته شده‌اند. این پروتکلها بعلت انتشار سریع (که اغلب دلیلی جز مجانی بودن ندارد) بسرعت جا می‌افتند، و بهمین دلیل جایگزین کردن آنها بسیار دشوار می‌شود. برخی از این پروتکلها امروز چیزی جز شرمساری نیستند. مثلاً، پروتکل TELNET اساساً برای ترمینالهای کُند و سنگین تله‌تایپ نوشته شد، و هیچ نشانی از گرافیک و ماوس در آن نیست. اما، بعد از ۲۵ سال خیلی‌ها همچنان از آن استفاده می‌کنند.

بطور خلاصه، مدل OSI (علیرغم برخی از مشکلات آن، و منهای لایه‌های نشست و نمایش) ثابت کرده که بهترین ابزار برای توصیف شبکه‌های کامپیوتریست - اما متأسفانه همچنان روی کاغذ باقی مانده است. از طرف دیگر، با اینکه چیزی بنام مدل TCP/IP در واقع **وجود خارجی ندارد، اما پروتکلهای** آن در مقیاس وسیعی مورد استفاده قرار می‌گیرند. از آنجائیکه در دنیای کامپیوتر هر کسی حرف خود را می‌زند، ما هم در این کتاب از یک مدل اصلاح شدۀ OSI استفاده خواهیم کرد، ولی توجه خود را بیشتر روی TCP/IP و پروتکلهای وابسته به آن معطوف می‌کنیم - البته دربارۀ پروتکلهای جدید مانند 802 ، SONET و بلوتوث هم صحبت خواهیم کرد. در حقیقت، مدلی که ما در این کتاب از آن استفاده خواهیم کرد، یک مدل ترکیبی (شکل ۱-۲۴) است.

5	لایه کاربرد
4	لایه‌انتقال
3	لایه شبکه
2	لایه پیوند داده
1	لایه فیزیکی

شکل ۱-۲۴. یک مدل مرجع ترکیبی، که در این کتاب از آن استفاده خواهد شد.

۵ـ۱ شبکه‌های نمونه

بحث شبکه‌های کامپیوتری انواع مختلفی از شبکه‌ها (کوچک و بزرگ، شناخته شده و مهجور) را در بر می‌گیرد. این شبکه‌ها در اندازه، کاربرد و تکنولوژیهای بکار رفته با هم متفاوتند. برای آن که تصوری از تنوع موجود در شبکه‌های کامپیوتری داشته باشید، در این قسمت نگاهی به چند نمونه از آنها خواهیم انداخت.

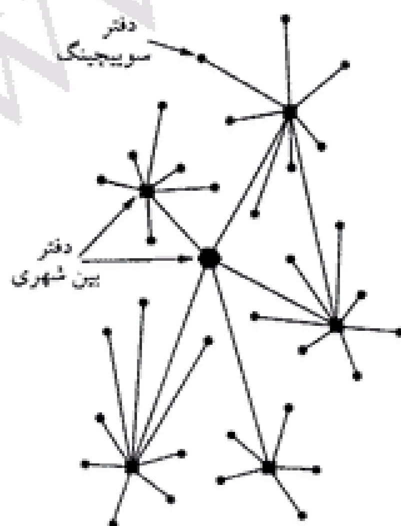
این قسمت را با معرفی اینترنت (که احتمالاً شناخته‌شده‌ترین شبکه‌هاست) شروع می‌کنیم، و نگاهی به تاریخچه، سیر تکامل و تکنولوژی آن خواهیم داشت. پس از آن به سراغ ATM (که تفاوتهای چشمگیری با اینترنت دارد، و حتی می‌توان گفت ضد آن است) می‌رویم. و بالاخره، نگاهی به IEEE 802.11 (استاندارد شبکه‌های محلی بیسیم) می‌اندازیم.

۱ـ۵ـ۱ اینترنت

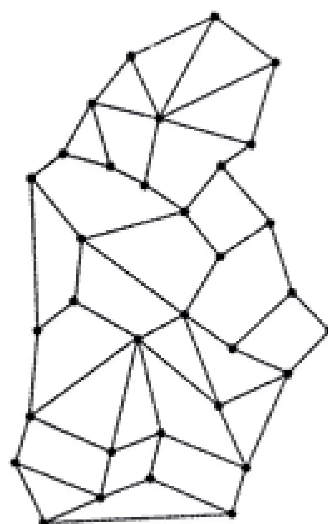
اینترنت *(Internet)* در واقع اصلاً یک شبکه نیست، بلکه مجموعه‌ایست از شبکه‌های مختلف که از پروتکلهای خاصی استفاده کرده، و سرویسهای مشخصی را ارائه می‌کند. ویژگی غیرعادی اینترنت اینست که توسط فرد خاصی طراحی نشده، و هیچکس هم آنرا کنترل نمی‌کند. برای درک بهتر این مطلب، اجازه دهید ببینیم اینترنت از کجا شروع شد، و علت آن چه بود. یکی از جالبترین تاریخچه‌های اینترنت را می‌توانید در کتاب **جان نافتون** - 2000 - ببینید. این کتاب نه تنها برای افراد عادی، بلکه برای مورخان نیز جلب است؛ برخی از مطالب ذیل از این کتاب اقتباس شده است. البته کتابهای فنی بیشماری نیز دربارهٔ اینترنت و پروتکلهای آن نوشته شده، که از آن میان می‌توان به (Maufer, 1999) اشاره کرد.

آرپانت (ARPANET)

داستان ما از اواخر دهۀ ۱۹۵۰ شروع می‌شود. در اوج جنگ سرد، وزارت دفاع ایالات متحدۀ آمریکا به فکر ایجاد یک شبکۀ فرماندهی و کنترل افتاد که بتواند حتی در مقابل حملات هسته‌ای دوام بیاورد. در آن زمان تمامی مخابرات نظامی به شبکۀ تلفن عمومی متکی بود، که مستعد آسیب تشخیص داده شده بود. با یک نگاه به شکل ۱-۲۵ (الف) می‌توانید مبنای این استدلال را دریابید. در این شکل نقاط سیاه نمایندۀ مراکز سوئیچینگ شهری



(الف)


(ب)

شکل ۱-۲۵. (الف) ساختار شبکۀ تلفن. (ب) طرح بارن برای یک سیستم سوئیچینگ توزیع شده.

هستند که هزاران خط تلفن از آنها منشعب می‌شود. این مراکز نیز بنوبهٔ خود به مراکز بین شهری بزرگتر متصل‌هستند، که در مجموع شبکهٔ تلفن کشوری را می‌سازند. آسیب‌پذیری این سیستم از آنجا ناشی می‌شد که تخریب چند مرکز بین‌شهری کلیدی می‌توانست تماس تلفنی را در کل کشور مختل کند.

در سال ۱۹۶۰ وزارت دفاع قراردادی را با **شرکت راند** (RAND Corporation) امضا کرد، که در آن وظیفهٔ یافتن یک راه‌حل به آن محول شده بود. یکی از متخصصان این شرکت، بنام **پُل بارن** (Paul Baran) ، طرح یک شبکه **توزیع شده** (distributed) و **تحمل‌پذیر خطا** (fault-tolerant) را پیشنهاد کرد، که آنرا در شکل ۱-۲۵ (ب) می‌بینید. از آنجائیکه در این شبکه طول مسیر بین مراکز سوئیچینگ طولانیتر از آن بود که بتوان از سیگنالهای آنالوگ استفاده کرد، بارن پیشنهاد کرد در این سیستم از تکنولوژی **سوئیچینگ بستهٔ دیجیتالی** (digital packet-switching) استفاده شود. بارن گزارشات متعددی برای وزارت دفاع نوشت، و جزئیات سیستم پیشنهادی خود را تشریح کرد. مقامات رسمی پنتاگون به ایدهٔ نهفته در این سیستم علاقمند شدند، و از AT&T (که در آن زمان انحصار شبکهٔ تلفن کشوری را در دست داشت) خواستند که یک نمونهٔ اولیه از آن بسازد. AT&T طرح بارن را رد کرد؛ بزرگترین و ثروتمندترین شرکت دنیا تحمل نمی‌کرد که یک جوان تازه از راه رسیده به آنها بگوید چگونه شبکهٔ تلفن بسازند! آنها ادعا کردند که طرح بارن قابل اجرا نیست، و بدین ترتیب ایدهٔ آن را در نطفه خفه کردند.

سالها گذشت، و وزارت دفاع همچنان بدنبال سیستم فرماندهی و کنترل ایده‌آل خود بود. برای درک بهتر اتفاقات بعدی، باید کمی به عقب برگردیم: به اکتبر ۱۹۵۷، زمانی که اتحاد جماهیر شوروی (سابق) با پرتاب اولین قمر مصنوعی بنام **اسپوتنیک** در مسابقهٔ فضایی از ایالات متحده پیشی گرفت. آیزنهاور، رئیس جمهور وقت ایالات متحده، در جستجو برای یافتن علت عقب‌افتادگی کشورش، با وحشت دریافت که نیروهای زمینی، دریایی و هوایی آمریکا مشغول دعوا بر سر تقسیم بودجهٔ تحقیقاتی پنتاگون هستند. وی بلافاصله تصمیم گرفت که یک مرکز واحد برای تحقیقات نظامی بوجود آورد؛ مرکزی که **آرپا (آژانس پروژه‌های تحقیقاتی پیشرفته** (Advanced Research Projects Agency - ARPA : نام گرفت. آرپا هیچ دانشمند یا آزمایشگاهی نداشت؛ در واقع، آرپا چیزی نبود جز یک دفتر هماهنگی کوچک با بودجه‌ای ناچیز (البته با معیارهای پنتاگون). آرپا کارش را با عقد قرارداد یا واگذاری امتیاز به شرکتها یا دانشگاههایی که ایده‌های جالبی داشتند، انجام می‌داد.

در سالهای اول، آرپا بیشتر سعی داشت خطوط کلی مأموریت خود را روشن و ترسیم کند، ولی در سال ۱۹۶۷ توجه مدیرعامل آن، **لاری رابرتس**، به موضوع شبکه جلب شد. او با متخصصان بسیاری مشورت کرد؛ و یکی از همین متخصصان، بنام **وسلی کلارک**، بود که پیشنهاد ایجاد یک زیرشبکهٔ سوئیچینگ بسته را مطرح کرد (شکل ۱-۱۰).

بعد از مقداری بحثهای اولیه، رابرتس ایده را پسندید و آنرا طی یک مقالهٔ نسبتاً مبهم به **گردهمایی اصول سیستم عامل** (که در اواخر ۱۹۶۷ در گاتلین‌بورگ، تنسی برگزار شده بود) ارائه کرد (Roberts, 1967) . در میان ناباوری رابرتس، مقالهٔ دیگری نیز به این کنفرانس ارائه شده بود که نه تنها سیستم مشابهی را توصیف می‌کرد، بلکه حتی صحبت از پیاده‌سازی آن تحت مدیریت فردی بنام **دونالد دیویس** در **آزمایشگاه ملی فیزیک** (NPL) در انگلستان بمیان آمده بود. سیستم NPL در واقع سیستمی در سطح ملی نبود، بلکه فقط چند کامپیوتر را در محوطهٔ NPL به هم متصل می‌کرد، اما نکته مهم این بود که نشان می‌داد سوئیچینگ بسته در عمل کار می‌کند. از همه جالبتر اینکه، سیستم NPL بر اساس کارهای بارن پایه‌گذاری شده بود. وقتی رابرتس از گاتلین‌بورگ برگشت، دیگر مصمم بود چیزی را بسازد که بعدها به **آرپانت** (ARPANET) معروف شد.

این زیرشبکه تعدادی مینی‌کامپیوتر بنام IMP (Interface Message Processor) را با خطوط انتقال

56-kbps به هم متصل می‌کرد. برای رسیدن به قابلیت اعتماد بالا، هر IMP به حداقل دو IMP دیگر متصل می‌شد. این زیرشبکه در واقع یک **زیرشبکهٔ دیتاگرام** (datagram subnet) بود، بنابراین اگر تعدادی از خطوط یا IMP ها از بین می‌رفتند، پیامها می‌توانستند از طریق مسیرهای جایگزین به مقصد برسند.

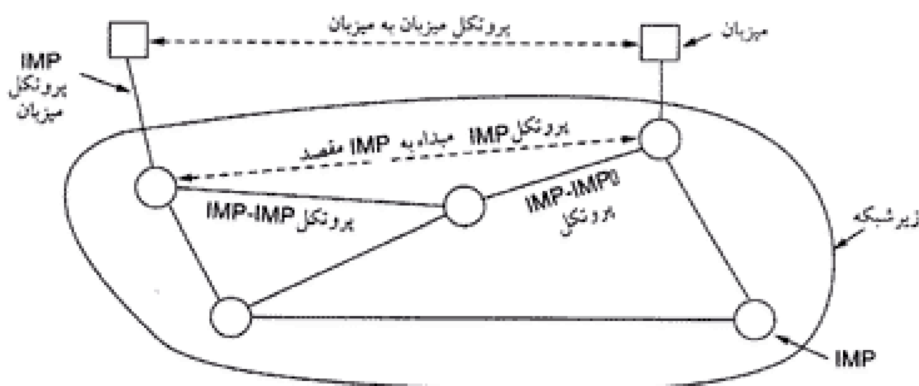
هر **گره** (node) این شبکه عبارت بود از یک کامپیوتر میزبان و یک IMP ، که با سیمی کوتاه به هم وصل می‌شدند. هر میزبان می‌توانست پیامهایی تا سقف ۸۰۶۳ بیت به IMP خود بفرستد، و این IMP سپس پیام را به بسته‌های ۱۰۰۸ بیتی شکسته و آنها را بصورت مستقل به سمت مقصد می‌فرستاد. هر بسته قبل از اینکه به گره بعدی هدایت شود، بایستی بطور کامل دریافت می‌شد؛ بدین ترتیب، آرپانت اولین زیرشبکهٔ سوئیچینگ بسته بود که بصورت **ذخیره-هدایت** (store-and-forward) کار می‌کرد.

پس از آن آرپا مناقصه‌ای برای ساخت این زیرشبکه اعلام کرد، که دوازده شرکت اسناد آنرا خریدند. بعد از بررسی پیشنهادات رسیده، در دسامبر ۱۹۶۸ آرپا شرکت BBN را (که یک شرکت مشاوره در کمبریج، ماساچوست بود) برای ساخت این زیرشبکه و نوشتن نرم‌افزارهای آن برگزید. شرکت BBN مینی‌کامپیوترهای اصلاح‌شدهٔ هانی‌ول DDP-316 را (که ۱۲ کیلوبایت حافظهٔ ۱۶ بیتی داشت) بعنوان IMP انتخاب کرد. از آنجائیکه قطعات مکانیکی ذاتاً غیر قابل اعتماد فرض می‌شدند، این IMP ها اصلاً دیسک نداشتند، و با خطوط اجاره‌ای 56-kbps به هم متصل می‌شدند. با اینکه امروزه حتی بچه‌ها هم دیگر خطوط 56-kbps را قبول ندارند (و به کمتر از ADSL راضی نمی‌شوند)، آنروزها خطوط 56-kbps بالاترین چیزی بود که می‌شد آرزو کرد.

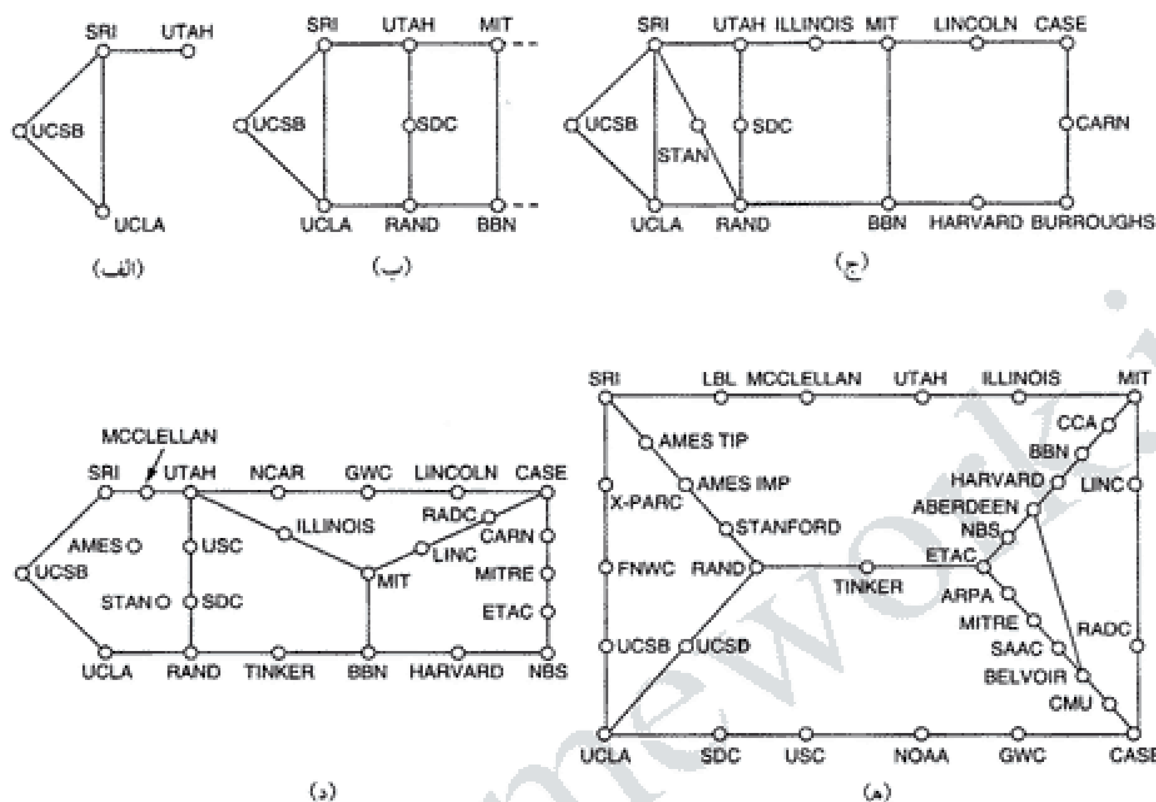
نرم‌افزار نیز در دو بخش مجزا طراحی شد: زیرشبکه، میزبان. نرم‌افزار زیرشبکه عبارت بود از پروتکل ارتباط IMP با میزبان، پروتکل IMP-IMP ، و نرم‌افزاری برای بهبود ارتباط IMP مبدأ با IMP مقصد. در شکل ۱-۲۶ طراحی اولیهٔ آرپانت را ملاحظه می‌کنید.

در خارج از زیرشبکه، میزبانها نیز به نرم‌افزار نیاز داشتند: پروتکل ارتباط میزبان با IMP ، پروتکل میزبان-میزبان، و نرم‌افزارهای کاربردی. بزودی معلوم شد که BBN احساس می‌کند با گرفتن پیام در نقطهٔ واسط میزبان-IMP ، و تحویل آن در نقطهٔ IMP-میزبان سمت مقابل کارش پایان یافته است.

اما رابرتس مشکل دیگری داشت: کامپیوترهای میزبان هم نیازمند نرم‌افزار بودند. برای حل این مشکل، در تابستان ۱۹۶۹ رابرتس همایشی از متخصصان شبکه (که عمدتاً دانشجویان تازه فارغ‌التحصیل بودند) در اسنوبِرد، یوتا تشکیل داد. این دانشجویان فکر می‌کردند کسی وجود دارد که طرح کلی شبکه را برای آنها توضیح دهد، و بعد از آن می‌توانستند نوشتن نرم‌افزار را شروع کنند. آنها بسیار شگفت‌زده شدند وقتی فهمیدند که نه متخصصی برای توضیح طرح شبکه وجود دارد، و نه اساساً چیزی بنام طرح شبکه! آنها دریافتند که باید کار را از صفر شروع کنند.



شکل ۱-۲۶. طراحی اولیهٔ آرپانت.



شکل ۱-۲۷. مراحل رشد آرپانت. (الف) دسامبر ۱۹۶۹. (ب) ژوئیه ۱۹۷۰. (ج) مارس ۱۹۷۱. (د) آوریل ۱۹۷۲. (ه) سپتامبر ۱۹۷۲.

با وجود همهٔ این مشکلات، بالاخره آرپا موفق شد در دسامبر ۱۹۶۹ یک شبکهٔ آزمایشی متشکل از چهار گره (دانشگاههای UCSB ، UCLA ، SRI و یوتا) راه‌اندازی کند. علت انتخاب این چهار دانشگاه آن بود که همگی آنها قرادادهای متعددی با آرپا داشتند، و از طرف دیگر (صرفاً برای زورآزمایی فنی) کامپیوترهای آنها بکلی با هم ناسازگار بود. با نصب IMP های جدید این شبکه گسترش یافت، و بزودی سراسر ایالات متحده را تحت پوشش گرفت. در شکل ۱-۲۷ رشد آرپانت را در طی سه سال پس از تولد آن ملاحظه می‌کنید.

آرپا برای کمک به رشد این نوزاد تازه متولد شده (آرپانت)، در زمینهٔ شبکه‌های ماهواره‌ای و مخابرات رادیویی نیز سرمایه‌گذاریهایی انجام داد. در یک آزمایش معروف، با استفاده از یک شبکهٔ رادیویی پیامهایی از یک کامیون در حال حرکت در جاده‌های کالیفرنیا به دانشگاه SRI ، و از آنجا از طریق آرپانت به ساحل شرقی ایالات متحده فرستاده شد، که سپس از آنجا از طریق شبکهٔ ماهواره‌ای به دانشگاه کالج در لندن هدایت شد. بدین ترتیب محققانی که در کامیونی در جاده‌های کالیفرنیا نشسته بودند، توانستند با کامپیوترهایی در لندن کار کنند.

این آزمایش همچنین نشان داد که پروتکلهای موجود آرپانت برای کار روی شبکه‌های مختلف مناسب نیستند. این نتایج منجر به تحقیقات بیشتر روی پروتکلها شد، که با اختراع TCP/IP و پروتکلهای آن به اوج رسید (Cerf and Kahn, 1974) . مدل TCP/IP بویژه برای ارتباطات روی شبکه‌های مختلف و ناهمگن (که آرپانت روز بروز به سمت آن حرکت می‌کرد) طراحی شده بود.

بمنظور تشویق و ترغیب پذیرش این پروتکلهای جدید، آرپا قراردادهایی با شرکت BBN و دانشگاه کالیفرنیا در برکلی (UCB) منعقد کرد، تا این پروتکلها را با یونیکس برکلی یکپارچه کنند. محققان برکلی هم کار خود را با

نوشتن برنامه‌های واسط شبکه (که به **سوکت** – socket – معروف شدند)، و برنامه‌های کاربردی و مدیریتی بنحو احسن انجام دادند.

زمانه نیز با TCP/IP یار بود؛ بسیاری از دانشگاهها تازه کامپیوترهای جدید VAX را خریده، و آنها را در شبکه‌های LAN به هم متصل کرده بودند، اما هیچ نرم‌افزاری برای شبکه کردن آنها نداشتند. وقتی یونیکس 4.2BSD (با پروتکلهای TCP/IP ، سوکتها و نرم‌افزارهای کمکی خود) بعنوان یک بستهٔ نرم‌افزاری کامل به بازار آمد، بلافاصله مورد قبول جامعهٔ دانشگاهی قرار گرفت. از همه مهمتر اینکه، با TCP/IP می‌شد به آرپانت وصل شد، اتفاقی که بسیاری منتظر آن بودند.

در دههٔ ۱۹۸۰ شبکه‌های بسیاری (بویژه شبکه‌های محلی) به آرپانت ملحق شدند. با افزایش تعداد کامپیوترهای آرپانت مشکل جدیدی پدید آمد، و آن پیدا کردن یک کامپیوتر در میان خیل عظیم کامپیوترها بود. برای حل این مشکل **سیستم نام ناحیه** (Domain Name System - DNS) ابداع شد، که نام کامپیوترها را به آدرس IP آنها تبدیل می‌کرد. از آن به بعد، DNS تبدیل به یک پایگاه دادهٔ عمومی و توزیع شده شد، که علاوه بر آدرس IP کامپیوترها، اطلاعات دیگری را نیز در اختیار کاربران خود قرار می‌داد. در فصل ۷ دربارهٔ DNS مفصلاً صحبت خواهیم کرد.

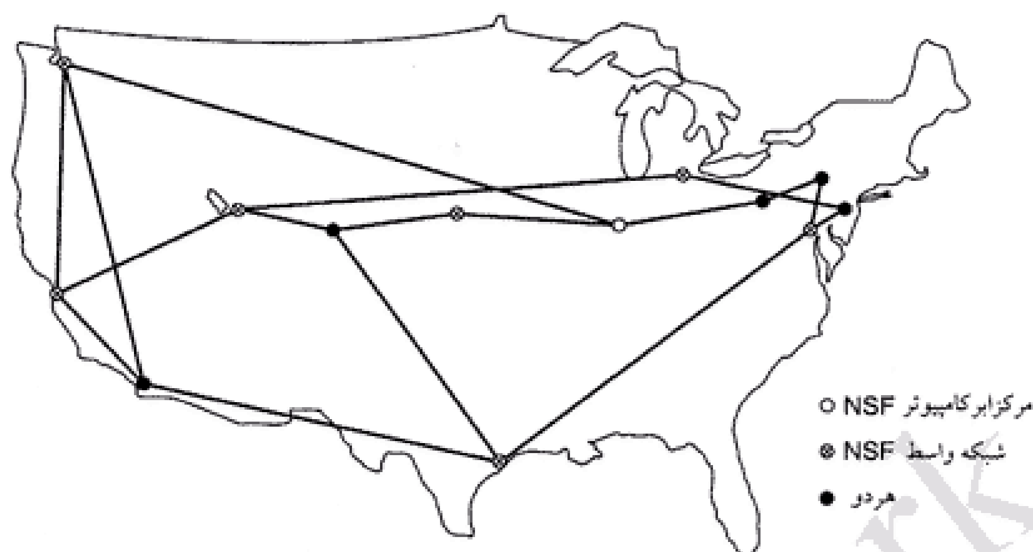
NSFNET

در اواخر دههٔ ۱۹۷۰ **بنیاد ملی علوم ایالات متحده** (U.S. National Science Foundation - NSF) شاهد تأثیر روزافزون آرپانت بر تحقیقات دانشگاهی بود. اما هر دانشگاهی که می‌خواست به آرپانت دسترسی داشته باشد، بایستی قراردادی با وزارت دفاع داشته باشد (که بسیاری از آنها نداشتند). پاسخ NSF به این وضعیت، راه‌اندازی شبکه‌ای مشابه آرپانت بود که تمام دانشگاهها به آن دسترسی داشته باشند. بمنظور ایجاد زیربنایی محکم برای این شبکه، NSF با متصل کردن شش ابرکامپیوتر خود در دانشگاههای سان‌دیه‌گو، بولدر، کامپاین، پیتسبورگ، ایتاکا و پرینستون، یک **ستون فقرات** (backbone) بوجود آورد. هر یک از این ابرکامپیوترها یک برادر کوچکتر (یک مینی‌کامپیوتر LSI 11 ، معروف به **فازبال**) داشت. این فازبال‌ها به خطوط اجاره‌ای 56-kbps متصل بودند، و زیرشبکه را می‌ساختند – یعنی، شبکهٔ NSF از نظر سخت‌افزاری شبیه آرپانت بود. اما، تکنولوژی نرم‌افزاری آن با آرپانت متفاوت بود: فازبال‌ها از همان ابتدا به TCP/IP صحبت می‌کردند، که آنرا تبدیل به این اولین شبکهٔ گستردهٔ TCP/IP می‌کرد.

بعدها NSF تعداد زیادی شبکهٔ منطقه‌ای تأسیس کرد، که به هزاران دانشگاه، آزمایشگاه تحقیقاتی، کتابخانه، و موزه اجازه می‌داد تا به هر یک از ابرکامپیوترهای آن دسترسی داشته باشند، یا اینکه مستقیماً با یکدیگر تماس برقرار کنند. این شبکه (شامل ستون فقرات و شبکه‌های محلی) NSFNET نامیده شد. از طریق لینکی بین یک IMP و یک فازبال در دانشگاه کارنگی-ملون، NSFNET به آرپانت نیز متصل شده بود. اولین ستون فقرات NSFNET را در شکل ۱-۲۸ مشاهده می‌کنید.

NSFNET یک موفقیت آنی بود، و از همان ابتدا با تراکم کاری روبرو شد. NSF بلافاصله به فکر گسترش NSFNET افتاد، و به همین منظور قراردادی با کنسرسیوم MERIT بست. برای ایجاد دومین ستون فقرات، کانالهای فیبرنوری با ظرفیت 448-kbps از MCI (که اکنون در WorlCom ادغام شده است) اجاره شد. برای مسیریاب‌های شبکه نیز از IBM PC-RT استفاده شد. این شبکه نیز بسیار زود با تراکم کاری روبرو شد، و در سال ۱۹۹۰ ظرفیت ستون فقرات آن به 1.5-Mbps ارتقاء داده شد.

با ادامهٔ رشد NSFNET ، بزودی NSF متوجه شد که دولت نمی‌تواند برای همیشه به سرمایه‌گذاری در شبکه ادامه دهد. از طرف دیگر، شرکتهای تجاری نیز مایل بودند به شبکهٔ NSFNET ملحق شوند، ولی مقررات NSF



شکل ۱-۲۸. ستون فقرات NSFNET در سال ۱۹۸۸.

کاربردهای انتفاعی شبکه را ممنوع کرده بود. متعاقب آن، NSF بعنوان اولین قدم بسوی تجاری کردن شبکه، شرکتهای IBM ، MERIT و MCI را به ایجاد یک مؤسسهٔ غیرانتفاعی (بنام Advanced Networks and Services - ANS) ترغیب کرد. در سال ۱۹۹۰، ANS کنترل NSFNET را بدست گرفت، و با ارتقاء لینکهای 1.5-Mbps به 45-Mbps شبکهٔ ANSNET را بوجود آورد. این شرکت بعد از ۵ سال کار به AOL (America Online) فروخته شد. اما در آن زمان دیگر شرکتهای بسیاری سرویسهای تجاری IP ارائه می‌کردند، و روشن شده بود که دولت باید پای خود را از تجارت شبکه بیرون بکشد.

برای تسهیل امور (و اطمینان از اینکه تمام شبکه‌های منطقه‌ای می‌توانند با هم تماس بگیرند)، NSF چهار قرارداد با شرکتهای بزرگ برای ایجاد **نقطهٔ دسترسی شبکه** (Network Access Point - NAP) امضا کرد. این چهار شرکت عبارت بودند از: PacBell (در منطقهٔ سانفرانسیسکو)، Ameritech (در منطقهٔ شیکاگو)، MFS (در منطقهٔ واشینگتن دی.سی.)، Sprint (در منطقهٔ نیویورک). هر اپراتور شبکه که بخواهد سرویسهای ستون فقرات به شبکه‌های منطقه‌ای NSF بدهد، بایستی به تمام NAP ها متصل باشد.

بدین ترتیب، هر بسته که بخواهد از یک منطقه به منطقهٔ دیگر برود، می‌تواند از هر یک از این ستونهای فقرات استفاده کند، که نتیجهٔ آن ایجاد رقابت برای سرویس بهتر و قیمت کمتر است. با این تمهید، ستون فقرات منحصربفرد دولتی جای خود را به یک زیرساخت متنوع و رقابتی داد. بسیاری از افراد دولت فدرال را به گناه عدم خلاقیت سرزنش می‌کنند، ولی در واقع این بنیاد ملی علوم و وزارت دفاع بودند که زیرساخت‌های اینترنت را شکل داده و سپس ادارهٔ آنرا به بخش خصوصی سپردند.

در دههٔ ۱۹۹۰ مناطق و کشورهای بسیاری، با تأثیرپذیری از الگوی آرپانت و NSFNET ، شبکه‌های ملی تحقیقاتی خود را بوجود آوردند. در اروپا، این شبکه‌ها (که EuropaNET و EBONE نام داشتند) از لینکهای 2-Mbps شروع کردند، و به 34-Mbps ارتقاء یافتند. در آنجا نیز زیرساخت‌های شبکه بتدریج به بخش خصوصی محول شد.

کاربردهای اینترنت

بعد از آنکه در اول ژانویه ۱۹۸۳ TCP/IP بعنوان تنها پروتکل رسمی آرپانت معرفی شد، تعداد شبکه‌ها، کامپیوترها و کاربران متصل به آن بسرعت افزایش یافت؛ و وقتی آرپانت و NSFNET به هم متصل شدند، رشد آن

حالت نمایی بخود گرفت. بسیاری از مناطق و کشورها (از جمله کانادا، اروپا و اقیانوسیه) به شبکه ملحق شدند.

در اواسط دهۀ ۱۹۸۰ دیگر افراد به این مجموعه به عنوان شبکه‌ای از شبکه‌ها (که بعدها به **اینترنت** معروف شد) نگاه می‌کردند، بدون آنکه هیچگونه بخشنامۀ رسمی در کار باشد، یا حتی مراسم افتتاحیه‌ای (با قیچی و نوارهای رنگی، و تشویق و هورا) برگزار شده باشد.

چسبی که اینترنت را به هم متصل نگه می‌دارد، مدل TCP/IP و مجموعه پروتکلهای آن است. پذیرش TCP/IP باعث شد تا سرویسهای جهانی بتوانند جنبۀ عملی بخود بگیرند.

اما واقعاً **«روی اینترنت بودن»** چه معنایی دارد؟ طبق تعریف ما، ماشینی روی اینترنت است که مجموعه پروتکلهای TCP/IP را اجرا کند، یک آدرس IP داشته باشد، و بتواند بسته‌های IP را به تمام ماشینهای دیگری که روی اینترنت هستند، بفرستد. صرف توانایی ارسال و دریافت ایمیل به معنای بودن روی اینترنت نیست، چون سرویسهای ایمیل می‌تواند به شبکه‌های خارج از اینترنت هدایت شود. با این حال، اوضاع با وضعیتی که در حال حاضر وجود دارد (میلیونها کامپیوتر شخصی می‌توانند با مودم به یک ISP وصل شده، یک آدرس IP موقتی بگیرند، و بسته‌های IP ردوبدل کنند)، کمی مغشوش و مبهم است. اما، مادامیکه این کامپیوترها به مسیریاب ISP متصل هستند، پُر بیراه نیست که آنها را روی اینترنت بدانیم.

اینترنت سنتی (از ۱۹۷۰ تا اوایل دهۀ ۱۹۹۰) چهار کاربرد عمده داشت:

۱. **ایمیل** (e-mail) - نوشتن، ارسال و دریافت نامه‌های پُست الکترونیک از همان روزهای اول آرپانت جزء سرویسهای آن بود، و همچنان یکی از محبوبترین‌هاست. امروزه بسیاری از افراد روزانه دهها و صدها ایمیل دریافت می‌کنند، و به آن بعنوان دریچه‌ای برای ارتباط با دنیای خارج نگاه می‌کنند - بسیار بیشتر از تلفن یا پُست معمولی.

۲. **اخبار** (news) - گروه خبری (newsgroup) یک محفل اختصاص یافته برای تبادل پیام در یک زمینۀ خاص است. امروزه هزاران گروه خبری در زمینه‌های فنی و غیرفنی (از جمله کامپیوتر، علوم، هنر و سیاست) وجود دارند. هر گروه خبری برای خود قواعد و مقرراتی دارد، که سرپیچی از آنها را برنمی‌تابد.

۳. **ورود از راه دور** (remote login) - هر روز هزاران نفر در سراسر دنیا برای ورود به کامپیوترهای دیگر از طریق اینترنت (البته آنهایی که حق ورود به آنها را داشته باشند)، از برنامه‌هایی مانند telnet ، rlogin یا ssh استفاده می‌کنند.

۴. **انتقال فایل** (file transfer) - با استفاده از برنامه‌های FTP ، کاربران اینترنت می‌توانند فایلهای خود را از یک ماشین به ماشین دیگر کپی کنند. جریان انتقال دانش از این طریق بسیار گسترده و متنوع است.

تا اوایل دهۀ ۱۹۹۰ اینترنت جولانگاه دانشگاهیان، کارمندان دولت و محققان صنعتی بود؛ اما یک برنامۀ کاربردی جدید بنام WWW (World Wide Web) این وضعیت را بکلی تغییر داد، و میلیونها نفر افراد عادی نیز توانستند به کاربران حرفه‌ای اینترنت ملحق شوند. این برنامه، که توسط **تام برنرز-لی** (Tom Berners-Lee) از فیزیکدانان مرکز تحقیقات هسته‌ای اروپا (CERN) ابداع شد، هیچ یک از سرویسهای اینترنت را عوض نکرد، ولی کاربرد آنها را ساده‌تر کرد. به کمک این تکنولوژی جدید، و برنامۀ **مرورگر موزائیک** (Mosaic browser) ، که توسط مارک آندرسن در مرکز ملی کاربردهای ابَرکامپیوتر (NCSA) نوشته شد، WWW ایجاد سایتهایی متشکل از صفحات مختلف (و با اطلاعاتی در قالبهای متن، تصویر، صدا، و حتی ویدئو)، با لینکهایی به صفحات دیگر را امکانپذیر کرد. با کلیک کردن روی یک **لینک** (link) ، کاربر مستقیماً به صفحه‌ای که مشخص شده، می‌پرد. حتماً سایتهای بسیاری متعلق به شرکتهای بزرگ را دیده‌اید، که می‌توانید با کلیک کردن هر یک از لینکهای آن، به صفحۀ مربوطه (مثلاً، صفحۀ مربوط به محصولات شرکت، لیست قیمتهای آن، پشتیبانی فنی، فروش و غیره) وارد شوید.

در زمانی کوتاه صفحات جدید و متنوعی به WWW اضافه شد، صفحاتی مانند نقشۀ شهرها و کشورها،

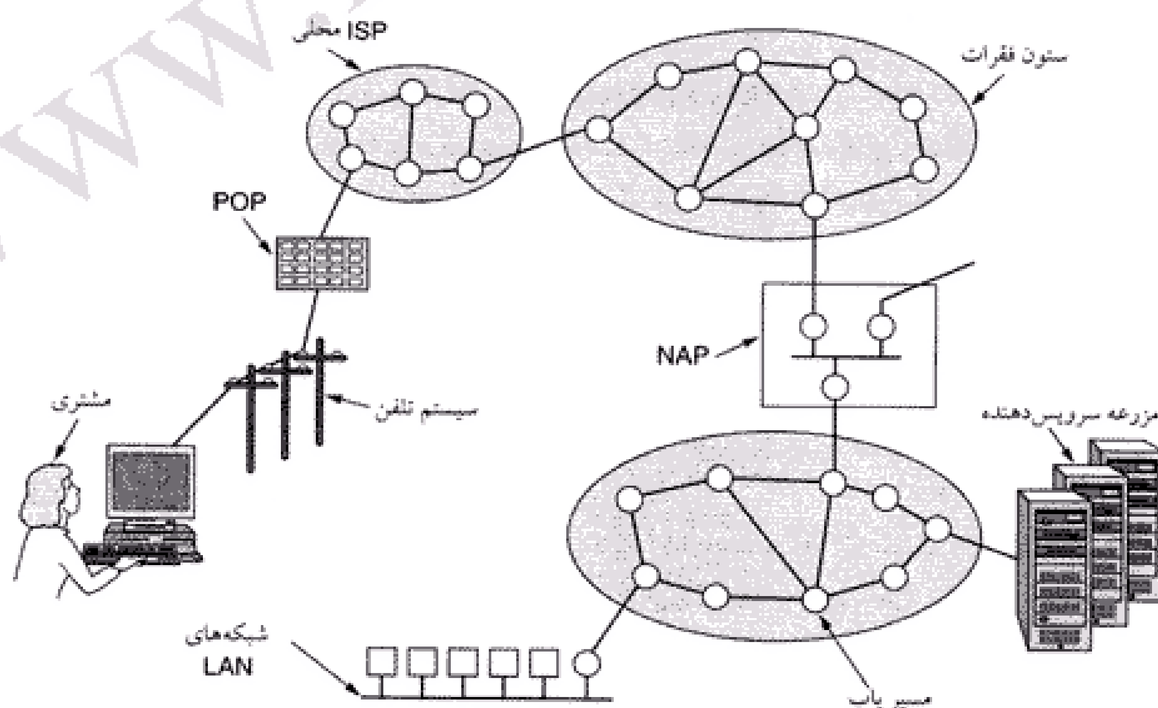
جدول قیمت سهام، کاتالوگ کارتهای کتابخانه‌ها، برنامه‌های رادیویی، و حتی متن کامل کتابهایی که از شمول قانون حق‌التألیف خارج شده‌اند (مانند کتابهای مارک تواین، چارلز دیکنز، و امثالهم). حتی بسیاری از افراد عادی نیز برای خود سایت (صفحات خانگی) ساخته‌اند.

موتور محرکهٔ این رشد، شرکتهای **ارائه‌دهندهٔ سرویس اینترنت** (Internet Service Provider - ISP) بودند. این شرکتها به افراد اجازه می‌دادند تا از خانه و با کامپیوترهای شخصی خود به اینترنت متصل شده، و از سرویسهای آن استفاده کنند. در دههٔ ۱۹۹۰، این شرکتها هر ساله برای دهها میلیون نفر امکان دسترسی اینترنت فراهم کردند، و چهرهٔ آنرا از محیطی دانشگاهی و نظامی به یک شبکهٔ عمومی تغییر دادند. تعداد دقیق کاربران اینترنت در حال حاضر معلوم نیست، ولی محققاً سر به صدها میلیون نفر می‌زند، و خیلی زود از مرز یک میلیارد خواهد گذشت.

معماری اینترنت

در این قسمت سعی می‌کنیم تصویری کلی از اینترنت بدست دهیم (شکل ۱-۲۹ را ببینید). به دلیل شباهتها و تداخل وظایف زیادی که بین شرکتهای مخابرات و ISP ها وجود دارد، امروزه اوضاع بسیار در هم و مغشوش است، و بسختی می‌توان گفت کی چکاره است - به همین دلیل توضیحات ذیل ساده‌تر از آن چیزیست که در واقعیت وجود دارد. اجازه دهید شکل ۱-۲۹ را جزء به جزء بررسی کنیم.

بهترین نقطه برای شروع، خانهٔ **مشتری** (client) است. در اینجا فرض را بر این گذاشته‌ایم که مشتری با استفاده از یک مودم و خط تلفن به ISP متصل می‌شود. **مودم** (modem) وسیله‌ایست که سیگنالهای دیجیتالی کامپیوتر را به سیگنالهای آنالوگ تبدیل می‌کند، تا این سیگنالها بتوانند بدون اعوجاج روی خطوط تلفن منتقل شوند. این سیگنالها در **نقطهٔ تماس** ISP (که به Point Of Presence - POP معروف است) مجدداً تبدیل به سیگنالهای دیجیتال شده، و وارد شبکهٔ منطقه‌ای ISP می‌شود. از این نقطه به بعد سیستم کاملاً دیجیتال است، و بر مبنای سوئیچ بسته کار می‌کند. اگر این ISP همان شرکت مخابرات باشد، POP در مرکز سوئیچینگ تلفن واقع



شکل ۱-۲۹. یک تصویر کلی از اینترنت.

خواهد بود؛ اما اگر ISP و شرکت مخابرات یکی نباشند، POP یک مرکز سوئیچینگ کوچک بین راهی خواهد بود، که از آنجا به شبکهٔ تلفن وصل می‌شود.

شبکهٔ منطقه‌ای هر ISP از چند مسیریاب، که به شهرهای مختلف تحت پوشش آن ISP سرویس می‌دهند، تشکیل می‌شود. اگر مقصد بستهٔ ارسال شده از مشتری یکی از کامپیوترهای واقع در همان شبکهٔ منطقه‌ای ISP باشد، بلافاصله به آن تحویل داده می‌شود. ولی اگر چنین نباشد، بسته به اپراتور ستون فقرات ISP داده خواهد شد.

اپراتور ستون فقرات (backbone operator) بالاترین نقطهٔ این زنجیره است (شرکتهای At&T و Sprint جزء اپراتورهای عمده هستند). هر اپراتور یک شبکهٔ بزرگ از ستونهای فقرات بین‌المللی (متشکل از هزاران مسیریاب که با فیبرهای نوری پُرسرعت به هم متصلند) را اداره می‌کند. شرکتهای بزرگ و آنهایی که سرویسهای میزبانی اینترنت ارائه می‌کنند، معمولاً مستقیماً به ستون فقرات متصل هستند. اپراتورهای ستون فقرات این نوع خدمات را تشویق می‌کنند، و برای آن تسهیلات ویژه‌ای بنام **کاریر کرایه‌ای** فراهم می‌آورند، که بعلت ارتباط نزدیک با ستون فقرات از سرعتهای بسیار بالایی برخوردارند.

اگر مقصد بستهٔ ارسال شده یکی از ISP های متصل به ستون فقرات باشد، به نزدیکترین مسیریاب فرستاده می‌شود و از آنجا بدست وی خواهد رسید. با این حال در دنیا ستونهای فقرات متعددی (با سرعتهای مختلف) وجود دارند، و احتمال دارد که این بسته وارد یکی از این شاهراههای رقیب شود. برای اینکه بسته‌ها بتوانند براحتی بین شاهراهها حرکت کنند، تمام آنها باید به یک NAP متصل باشند – NAP چیزی نیست بیش از اتاقی پُر از مسیریاب‌های متعدد (حداقل یکی به ازای هر شاهراه)، که در یک LAN ساده به هم متصل شده‌اند. شاهراههای بزرگ، غیر از اتصال از طریق NAP ، معمولاً بصورت مستقیم نیز به شاهراههای دیگر راه دارند (که به آن **ارتباط دوجانبه** گفته می‌شود). یکی از تناقضهای بزرگ اینترنت آن است که شرکتهایی که در انظار عموم با هم رقابت سخت دارند، در خفا با یکدیگر ارتباطات نزدیک و تنگاتنگ برقرار می‌کنند (Metz, 2001) .

این هم از مروری اجمالی بر اینترنت. البته در فصلهای آینده تک تک این اجزاء، الگوریتمها، و پروتکلها را مفصلاً بررسی خواهیم کرد. نکته‌ای که جالبست بدانید اینست که امروزه بسیاری از شرکتها ارتباطات داخلی شبکهٔ خود را بر اساس مدل و تکنولوژیهای اینترنت بنا می‌کنند، چیزی که به **اینترانت** (intranet) معروف است.

۱-۵-۲ شبکه‌های اتصال-گرا: X.25 ، Frame Relay و ATM

از همان اولین روزهایی که شبکه پا به عرصهٔ وجود گذاشت، جنگ بین طرفداران زیرشبکه‌های اتصال-گرا و شبکه‌های غیرمتصل (دیتاگرام) نیز شروع شد. مهمترین برگ برندهٔ طرفداران زیرشبکه‌های غیرمتصل همان آرپانت/اینترنت است. بیاد دارید که قصد اولیهٔ وزارت دفاع آمریکا از بنیانگذاری آرپانت، ایجاد شبکه‌ای بود که بتواند در مقابل ضربات هسته‌ای (و منهدم شدن بخش بزرگی از خطوط و تجهیزات انتقال) دوام بیاورد (در واقع، هدف اصلی این طرح بالا بردن ضریب تحمل خرابی شبکه بود). این رهیافت منجر به طراحی شبکه‌ای شد که در آن هر بسته راه خود را مستقل از بسته‌های دیگر طی می‌کند. بدین ترتیب، اگر تعدادی از مسیریاب‌های شبکه از مدار خارج شوند، مادامیکه شبکه بتواند مسیرهای جدید خود را از نو پیکربندی کند، در ارسال بسته‌ها از مبدأ به مقصد خللی پیش نخواهد آمد.

طرفداران زیرشبکه‌های اتصال-گرا معمولاً همان شرکتهای تلفن هستند. در این سیستم، آغازکنندهٔ ارتباط قبل از آنکه بتواند ارسال اطلاعات را شروع کند، بایستی منتظر برقراری ارتباط مستقیم با طرف مقابل بماند. این ارتباط فیزیکی در تمام طول تماس برقرار می‌ماند، و تمام بسته‌های اطلاعات از همین مسیر واحد عبور خواهند کرد. اگر هر یک از تجهیزات این مسیر به هر دلیلی از کار بیفتد، تماس قطع خواهد شد – چیزی که وزارت دفاع مسلماً نمی‌پسندد.

پس علت علاقهٔ شرکتهای تلفن به این سیستم چیست؟ دو دلیل اصلی این علاقه عبارتند از:

۱. کیفیت سرویس
۲. حسابرسی مصرف‌کنندگان

در شبکه‌های اتصال-گرا هر تماس مقداری از منابع زیرشبکه (از قبیل توان پردازشی مسیریاب‌ها) را بخود اختصاص می‌دهد، و در صورتیکه این منابع به حالت اشباع برسند، تماس جدید امکانپذیر نبوده، و کاربر با بوق اشغال روبرو خواهد شد. در این روش تماس‌ها (بدلیل اختصاص منابع کافی) از کیفیت بالایی برخوردار هستند. از طرف دیگر، اگر در شبکه‌های غیرمتصل تعداد زیادی بسته به یکباره وارد یک مسیریاب شوند، ممکنست برخی از آنها (بدلیل کمبود امکانات پردازشی) در داخل رواتر از بین بروند. البته فرستنده متوجه این نقص خواهد شد، و بسته‌های گمشده را از نو ارسال خواهد کرد، ولی همین موضوع باعث افت کیفیت شبکه (بویژه در مورد صدا و تصویر) می‌شود. لازم به گفتن نیست که شرکتهای تلفن بیش از هر چیز نگران کیفیت صدا هستند، و به همین دلیل همچنان زیرشبکه‌های اتصال-گرا را ترجیح می‌دهند.

دلیل دومی که شرکتهای تلفن سرویسهای اتصال-گرا را بیشتر می‌پسندند، امکان صدور صورتحساب برای مشترکان است (کاری که مدتهاست به آن عادت کرده‌اند). هزینهٔ تماسهای بین شهری و خارج از کشور معمولاً بر حسب مدت مکالمه محاسبه می‌شود (علت اتخاذ این روش هم بیشتر سادگی آن بوده است). اگر تماس مستقیمی بین دو طرف مکالمه برقرار نباشد، طبعاً این شرکتها نمی‌توانند برای مشترکان خود صورتحساب صادر کنند.

سیستمهای محاسبهٔ صورتحساب هزینهٔ بسیار سنگینی به شرکتهای تلفن تحمیل می‌کند. اگر یک شرکت تلفن بر اساس یک شارژ ثابت ماهیانه از مشترکان خود پول دریافت کند، علیرغم بالا رفتن مصرف مشترکان، می‌تواند پول زیادی صرفه‌جویی کند. اما عوامل زیادی (که بیشتر آنها هم سیاسی هستند) با اتخاذ این روش مخالفت می‌کنند. جالبست بدانید که در اغلب سیستمهای مشابه (مانند تلویزیونهای کابلی و برخی پارکهای تفریحی) از روش محاسبهٔ ثابت استفاده می‌شود. در این سیستمها هم امکان پرداخت به ازای مصرف وجود دارد، ولی بدلیل هزینهٔ بالای صدور صورتحساب معمولاً از آن اجتناب می‌شود.

به دلایل فوق، جای تعجب نیست که شرکتهای تلفن طرفدار زیرشبکه‌های اتصال-گرا باشند؛ اما تعجب‌برانگیز این است که، اینترنت هم دارد به همین سمت پیش می‌رود - البته با این استدلال که این کار باعث بالا رفتن کیفیت سرویسهای صدا و تصویر آن خواهد شد. اکنون اجازه دهید چند شبکهٔ اتصال-گرا را بهتر بشناسیم.

X.25 و Frame Relay

اولین شبکهٔ اتصال-گرا که وارد سرویس عمومی شد، شبکهٔ X.25 بود. این شبکه در اوایل دهۀ ۱۹۷۰، و در زمانی طراحی شد که شرکتهای تلفن بصورت انحصاری عمل می‌کردند، و هر کشور شبکهٔ ملی خاص خود را داشت. برای استفاده از X.25 ، ابتدا کامپیوتر مبدأ با ماشین مقصد تماس تلفنی برقرار می‌کرد. از آنجائیکه در آن واحد تماسهای مختلفی می‌توانست وجود داشته باشد، به هر تماس تلفنی یک شماره داده می‌شد. بسته‌های داده بسیار ساده بودند: یک سرآیند ۳ بایتی و بدنه‌ای متشکل از ۱۲۸ بایت. **سرآیند** (header) تشکیل می‌شد از یک شماره تماس ۱۲ بیتی، یک **شمارهٔ ترتیب بسته** (packet sequence number) ، یک **عدد تصدیق دریافت** (acknowledgement number) ، و چند بیت متفرقه. شبکه‌های X.25 به مدت نزدیک به یک دهه با موفقیتی نسبی کار کردند.

در دهۀ ۱۹۸۰ شبکه‌های X.25 جای خود را به نوع جدیدی از شبکه‌های اتصال-گرا بنام frame relay (رلهٔ فریم) دادند. این شبکهٔ جدید اساساً هیچ نوع کنترل خطا و کنترل جریانی نداشت، و بسته‌ها به همان ترتیب

دریافت در مقصد تحویل می‌شدند (البته اگر به مقصد می‌رسیدند). این سه خصوصیت (فقدان کنترل خطا، فقدان کنترل جریان، و تحویل ترتیبی بسته‌ها) شبکه‌های frame relay را بسیار شبیه یک LAN بزرگ می‌کند، و در واقع بزرگترین کاربرد آن هم همین است: اتصال چند LAN دور از هم، و ایجاد یک LAN بزرگ. شبکه‌های frame relay هم نسبتاً موفق بودند، و هنوز در برخی جاها از آنها استفاده می‌شود.

حالت انتقال آسنکرون (ATM)

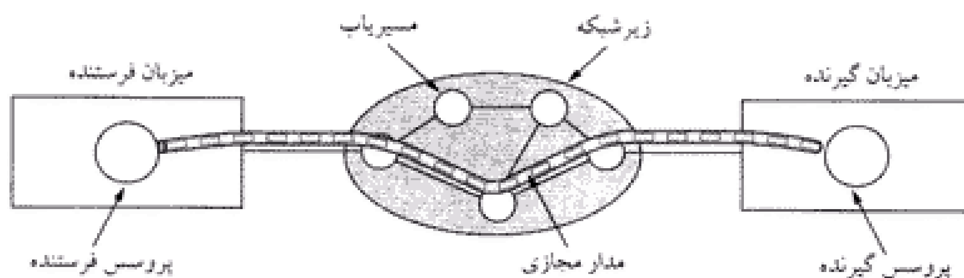
یکی دیگر از شبکه‌های اتصال-گرا (که اهمیت بسیار بیشتری نیز دارد) شبکهٔ **حالت انتقال آسنکرون** (Asynchronous Transfer Mode - ATM) است. علت این نامگذاری عجیب آن است که در شبکه‌های تلفن اکثر تماسها بصورت **سنکرون** (synchronous - وابسته به پالس ساعت) هستند، در حالیکه ATM چنین نیست.

شبکهٔ ATM در اوایل دههٔ ۱۹۹۰ طراحی شد، و سر و صدای زیادی نیز بپا کرد (Ginburg, 1996; Gorlaski, 1995; Ibe, 1997; Kime et al., 1994; Stallings, 2000). شبکه‌های ATM با ادعای ادغام تمام انواع شبکه و سیستمهای مخابراتی (صدا، داده، تلویزیون کابلی، تلکس، تلگراف، کبوترهای نامه‌بر، قوطیهای حلبی سیمی، طبلهای افریقایی، علامتهای دودی سرخپوستان، و خلاصه هر چیزی که به نوعی اطلاعات منتقل می‌کند) به میدان آمدند - اتفاقی که هرگز نیفتاد. علت آن هم تا تقریباً شبیه همان بلایی بود که سر OSI آمد (زمان نامناسب، تکنولوژی بد، پیاده‌سازی نامناسب، و سیاستهای غلط). شرکتهای اینترنتی که منتظر وسیله‌ای بودند تا شرکتهای تلفن را در همان راند اول از پا در آورند، به ATM امید بستند. اما این امید دیری نپائید، و شرکتهای اینترنتی (حتی سرسخت‌ترین آنها) بزودی دریافتند که تا رسیدن به سرویسهای مطلوب راه درازی در پیش دارند. البته ATM از OSI بسیار موفقتر بود، و حتی امروز هم در شبکه‌های تلفن (و برای انتقال بسته‌های IP) مورد استفاده قرار می‌گیرد. از آنجائیکه این زیرشبکه فقط برای ارتباطات داخلی بکار می‌رود، اغلب کاربران معمولی از وجود آن اطلاعی ندارد، ولی ATM زنده و سر حال است.

مدار مجازی ATM

از آنجائیکه شبکه‌های ATM از نوع اتصال-گرا هستند، برای برقراری ارتباط اولیه ابتدا باید یک بستهٔ خاص بفرستند. با عبور این بسته از زیرشبکه، تمام مسیریاب‌هایی که در مسیر آن قرار دارند، آنرا در جدولهای خود ثبت می‌کنند و منابع لازم را برای آن کنار می‌گذارند. به ارتباطی که بدین طریق برقرار می‌شود، **مدار مجازی** (virtual circuit) می‌گویند، چون بسیار شبیه مدارهای فیزیکی در شبکه‌های تلفن است (شکل ۱-۳۰ را ببینید). بسیاری از شبکه‌های ATM از مدارهای مجازی دائمی بین دو نقطه پشتیبانی می‌کنند (که بسیار شبیه خطوط اجاره‌ای در سیستم تلفن معمولی است). هر اتصال (موقت یا دائم) دارای یک شمارهٔ شناسایی است.

بعد از برقراری ارتباط، دو طرف می‌توانند شروع به فرستادن داده کنند. ایدهٔ اصلی در ATM ارسال داده‌ها در بسته‌های کوچک و با اندازهٔ ثابت، بنام **سلول** (cell) ، است. هر سلول ۵۳ بایت طول دارد، که ۵ بایت آن سرآیند، و



شکل ۱-۳۰. یک مدار مجازی.

بایت	5	48
	سرآیند	داده‌های کاربر

شکل ۱-۳۱. یک سلول ATM .

۴۸ بایت باقیمانده داده‌هاست (شکل ۱-۳۱). شمارهٔ شناسایی اتصال در سرآیند سلولها نوشته می‌شود، بطوریکه تمام مسیریاب‌های مسیر می‌توانند تشخیص دهند که هر سلول متعلق به کدام اتصال است، و چگونه باید آنرا هدایت کنند. هدایت سلولها بصورت سخت‌افزاری (و با سرعت فوق‌العاده بالا) صورت می‌گیرد - در واقع، علت اصلی اندازهٔ ثابت سلولها در شبکه‌های ATM اینست که ساخت مسیریاب‌های سخت‌افزاری برای آن بسیار ساده است (هدایت بسته‌های IP با اندازهٔ متغیر به مسیریاب‌های نرم‌افزاری نیاز دارد، که بسیار کُندتر هستند).

مزیت دیگر ATM توانایی آن در ارسال همزمان یک سلول به مسیرهای مختلف است - که این ویژگی در سیستمهای پخش تلویزیونی بسیار مفید است. از طرف دیگر، کوچک بودن سلولها باعث می‌شود تا هیچ خطی برای مدت طولانی اشغال نشود، و کیفیت سرویس افزایش یابد.

در ATM تمام سلولها از یک مسیر به مقصد هدایت می‌شوند. البته تضمینی برای رسیدن یک سلول به مقصد وجود ندارد، ولی ترتیب آنها حتماً رعایت می‌شود. اگر سلول ۲ بعد از سلول ۱ فرستاده شده باشد، به همان ترتیب به مقصد می‌رسند، و هرگز سلول ۲ پیش از سلول ۱ به مقصد نخواهد رسید. اگر هر یک از این سلولها (یا هر دوی آنها) در بین راه از بین بروند، این بر عهدهٔ پروتکلهای لایه‌های بالاتر است که آنها را بازیابی کنند. از این نظر ATM حداقل یک پله بالاتر از اینترنت می‌ایستد (که نه ترتیب بسته‌ها ضمانت می‌شود، نه حتی رسیدن صحیح و سالم آنها).

سازماندهی شبکه‌های ATM شبیه WAN های قدیمی (متشکل از خطوط تلفنی و سوئیچها) است. متداولترین سرعتها در شبکه‌های ATM عبارتند از 155-Mbps و 622-Mbps (البته ATM از سرعتهای بالاتر هم پشتیبانی می‌کند). علت انتخاب سرعت 155-Mbps آنست که **تلویزیونهای با وضوح بالا** (HDTV) به چنین سرعتی نیاز دارند - مقدار دقیق این سرعت 155.52-Mbps است، که دقیقاً معادل سرعت سیستم AT&T SONET می‌باشد. علت انتخاب سرعت 622-Mbps نیز اینست که از ترکیب چهار کانال 155.52-Mbps یک کانال 622-Mbps بوجود می‌آید.

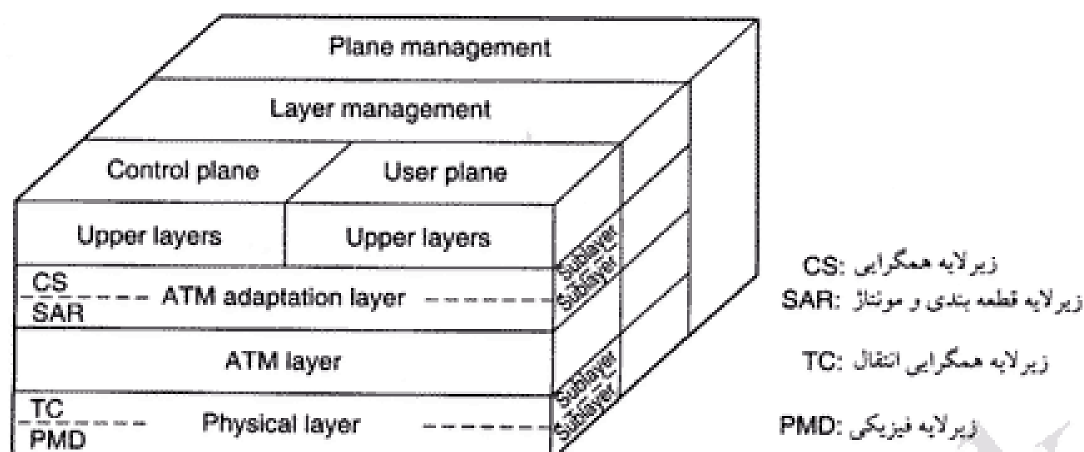
مدل مرجع ATM

شبکهٔ ATM برای خود یک مدل مرجع مستقل دارد، که با مدلهای OSI و TCP/IP فرق دارد - این مدل را در شکل ۱-۳۲ ملاحظه می‌کنید. این مدل سه لایه دارد: لایهٔ فیزیکی، لایهٔ ATM ، لایهٔ انطباق ATM (و هر چند لایه که کاربر مایل باشد روی این لایه‌ها سوار کند).

لایهٔ فیزیکی با مشخصات فیزیکی سیستم (ولتاژها، زمانبندی بیت‌ها و غیره) سر و کار دارد. مدل ATM هیچ پیشنیازی در مورد این مشخصات ندارد، و می‌گوید که ارسال سلولها می‌توانند بصورت مستقیم یا از طریق سیستمهای انتقال دیگر انجام شود. بعبارت دیگر، ATM مستقل از سیستم انتقال است.

لایهٔ ATM با خود سلولها و انتقال آنها سر و کار دارد. ایجاد و رها کردن مدار مجازی، تعریف فیلدهای سرآیند سلول، و **کنترل ازدحام** (congestion control) از وظایف این لایه است.

از آنجائیکه اکثر برنامه‌های کاربردی تمایلی به کار کردن با بسته‌هایی به کوچکی سلولهای ATM ندارند، یک لایهٔ دیگر بالای لایهٔ ATM تعبیه شده تا این قبیل برنامه‌ها بتوانند بسته‌های بزرگتری به ATM بفرستند. این لایه



شکل ۱-۳۲. مدل مرجع ATM .

(در سمت فرستنده) بسته‌های داده را به سلولهای ۵۳ بایتی می‌شکند، و در طرف گیرنده آنها را دوباره سر هم می‌کند. نام این لایه، **لایهٔ انطباق ATM** (ATM Adaptation Layer - AAL) است.
بر خلاف مدلهای قبلی که دو بُعدی بودند، مدل ATM یک مدل مرجع سه بُعدی است (شکل ۱-۳۲ را ببینید). **صفحهٔ کاربر** (user plane) با انتقال داده، کنترل جریان، تصحیح خطا، و دیگر عملکردهای کاربر سروکار دارد. از طرف دیگر، **صفحهٔ کنترل** (control plane) مدیریت اتصال را بر عهده دارد. وظیفهٔ صفحه‌های **مدیریت لایه** (layer management) و **مدیریت صفحه** (plane management) مدیریت منابع سیستم و هماهنگ کردن لایه‌های بینابینی است.
لایه‌های فیزیکی و AAL هر یک به دو زیرلایه تقسیم شده‌اند، یکی در پائین برای انجام عملکردهای محوله، و دیگری در بالا برای ارتباط با لایهٔ بالاتر (که **زیرلایهٔ همگرایی** - convergence sublayer - خوانده می‌شود). وظیفهٔ هر یک از این لایه‌ها و زیرلایه‌ها را در شکل ۱-۳۳ ملاحظه می‌کنید.

OSI لایه	ATM لایه	ATM زیرلایه	کارکرد
3/4	AAL	CS	واسط استاندارد
		SAR	قطعه‌بندی و مونتاژ
2/3	ATM		کنترل جریان تولید سرآیند سلول مدار مجازی - مدیریت مسیر مالتی پلکس/دیمالتی پلکس سلول .
2	Physical	TC	ایزوله کردن سرعت سلول تولیدمجموع تطبیقی تولید سلول بسته بندی وبازکردن بسته ها تولیدفریم
1		PMD	زمان بندی بیت دسترسی فیزیکی

شکل ۱-۳۳. وظایف لایه‌ها و زیرلایه‌های ATM .

زیرلایهٔ PMD (Physical Medium Dependent) مستقیماً به کابل شبکه وصل می‌شود، و کار آن ارسال ودریافت بیت‌ها و ایجاد همزمانی بین آنهاست. هر نوع کابل و سیستم انتقال زیرلایهٔ PMD خاص خود را دارد.

زیرلایهٔ دیگر لایهٔ فیزیکی، TC (Transmission Convergence) نام دارد. وقتی یک سلول ارسال می‌شود، لایهٔ TC آنرا بصورت جریانی از بیت‌ها به لایهٔ PMD می‌فرستد - که این کاری ساده است. در طرف گیرنده، لایهٔ TC باید جریان بیت‌هایی را که از لایهٔ PMD دریافت می‌کند، دوباره بصورت سلول در آورد - بعبارت دیگر باید بتواند ابتدا و انتهای هر سلول را بدرستی تشخیص دهد. در مدل ATM این کار در لایهٔ فیزیکی انجام می‌شود، وظیفه ی که در مدل OSI (و تقریباً تمام مدلهای دیگر) بر عهدهٔ لایهٔ پیوند داده است.

همانطور که قبلاً هم گفتیم، لایهٔ ATM مدیریت ایجاد و انتقال سلولها را بر عهده دارد. مهمترین بخش از وظایف ATM نیز در همین لایه صورت می‌گیرد. این لایه تلفیقی است از لایه‌های لینک داده و شبکه در مدل OSI - که در ضمن هیچ زیرلایه‌ای هم ندارد.

لایهٔ AAL به دو زیرلایهٔ SAR (Segmentation And Reassembly) و CS (Convergence Sublayer) تقسیم شده است. لایهٔ پائینی (SAR) در طرف فرستنده بسته‌های داده را به سلول می‌شکند، و در طرف گیرنده دوباره آنها را به هم می‌چسباند. لایهٔ بالایی (CS) وظیفهٔ ارائهٔ سرویسهای مختلف به برنامه‌های کاربردی را بر عهده دارد.

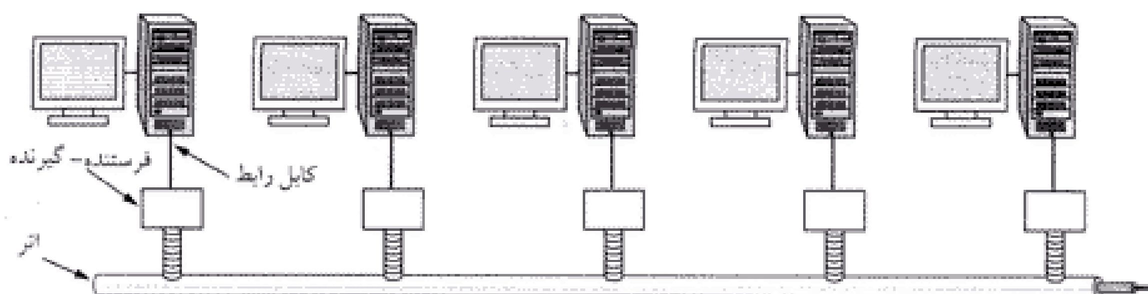
از آنجائیکه ATM در سراشیبی زوال قرار دارد، در این کتاب بیش از این دربارهٔ آن صحبت نخواهیم کرد. با این حال، بدلیل نصب در مقیاس وسیع، به احتمال زیاد تا چند سال دیگر نیز دوام خواهد آورد. برای کسب اطلاعات بیشتر دربارهٔ ATM به (Dobrowski and Grise, 2001; Gadecki and Heckart, 1997) مراجعه کنید.

۱-۵-۳ اترنت

اینترنت و ATM هر دو شبکه‌های گسترده هستند، ولی در هر شرکت، سازمان و دانشگاه تعداد زیادی کامپیوتر وجود دارد که باید به هم متصل شوند. از همین جاست که نیاز به شبکه‌های محلی شکل می‌گیرد. در این قسمت می‌خواهیم کمی دربارهٔ متداولترین شبکهٔ محلی، یعنی **اترنت** (Ethernet) ، صحبت کنیم.

داستان ما از ایالت بکر و دست نخوردهٔ هاوایی، در اوایل دههٔ ۱۹۷۰، شروع می‌شود - در اینجا منظور از «بکر و دست نخورده» فقدان شبکهٔ تلفن است. با اینکه نبود شبکهٔ تلفن برای کسانی که هاوایی را برای استراحت انتخاب می‌کردند، یک مزیت بود، ولی برای محققی بنام نورمن آبرامسون و همکارانش در دانشگاه هاوایی که می‌خواستند کاربران جزایر دورافتادهٔ هاوایی را به کامپیوتر مرکزی در هونولولو (مرکز ایالت هاوایی) متصل کنند، چندان خوشایند نبود. کشیدن کابل از وسط اقیانوس آرام مسلماً نمی‌توانست راه حل مشکل آنها باشد، پس باید فکر دیگری می‌کردند.

یکی از راه‌حل‌هایی که آنها پیدا کردند، استفاده از امواج رادیویی با بُرد کوتاه بود. هر ترمینال به یک رادیوی کوچک وصل می‌شد، که دو فرکانس داشت: فرکانس ارسال (به کامپیوتر مرکزی)، فرکانس دریافت (از کامپیوتر مرکزی). وقتی کاربر می‌خواست به کامپیوتر مرکزی وصل شود، روی فرکانس ارسال یک بسته می‌فرستاد. اگر کس دیگری در همان لحظه در حال ارسال نبود، بستهٔ مزبور به کامپیوتر مرکزی می‌رسید، و کاربر می‌توانست بستهٔ **تصدیق دریافت** (acknowledgement) آنرا روی فرکانس دریافت بگیرد. اما اگر کانال اشغال بود، ترمینال بستهٔ تصدیق دریافت را نمی‌گرفت، و متوجه می‌شد که باید دوباره سعی کند. از آنجائیکه روی کانال دریافت فقط یک کامپیوتر (کامپیوتر مرکزی) مجاز به ارسال اطلاعات بود، هرگز در آن انسداد پیش نمی‌آمد. این سیستم، که آلوهانت (ALOHANET) نام گرفت، در شرایط ترافیک پائین خوب کار می‌کرد، ولی در ترافیک بالا بشدت ناکارآمد می‌شد.



شکل ۱-۳۴. معماری اترنت اولیه.

در همان زمان، دانشجویی بنام **باب متکالف** که تازه از M.I.T فارغ‌التحصیل شده بود، به قصد ادامه تحصیل در مقطع دکترا وارد دانشگاه هاروارد شد. باب در حین مطالعات خود با کارهای آبرامسون آشنا، و بشدت به آن علاقمند شد. این علاقه تا آن حد بود که باب تصمیم گرفت قبل از شروع به کار در **مرکز تحقیقات زیراکس در پالوآلتو** (Xerox PARC)، تابستان را در هاوایی بگذراند و با آبرامسون کار کند. وقتی باب متکالف به Xerox PARC برگشت، متوجه شد که محققان آنجا روی پروژه‌ای کار می‌کنند که بعدها به کامپیوتر شخصی (Personal Computer - PC) معروف شد. ولی این یک ماشین ایزوله و جدا از همه جا بود. باب، با استفاده از تجارب آبرامسون، و به کمک یکی از همکارانش بنام دیوید باگز، اولین شبکهٔ محلی را طراحی و پیاده‌سازی کرد (Metcalfe and Boggs, 1976).

آنها این سیستم را (بیاد ماده‌ای خیالی بنام **اتر** - ether - که تا مدتها تصور می‌شد محیط انتشار امواج الکترومغناطیس است) **اترنت** نامیدند. (بعد از کشف معادلات انتشار امواج الکترومغناطیس توسط فیزیکدان بریتانیایی، جیمز کلارک ماکسول، دانشمندان فرض را بر این گذاشتند که این امواج در محیطی بنام اتر منتشر می‌شوند. فقط بعد از آزمایشات معروف مایکلسون-مورلی بود که فیزیکدانان دریافتند امواج الکترومغناطیس می‌توانند در خلاء منتشر شوند.)

رسانهٔ انتشار در این سیستم خلاء نبود، بلکه از یک رشته کابل **هم‌محور** *(coaxial)* ضخیم بطول حداکثر ۲/۵ کیلومتر (با یک تکرارکننده در هر ۵۰۰ متر) استفاده می‌شد (این کابل در واقع همان **اتر** محسوب می‌شد) - شکل ۱-۳۴ را ببینید. حداکثر تا ۲۵۶ کامپیوتر را می‌شد به این کابل متصل کرد. به چنین کابلی، که چندین ماشین بصورت موازی به آن متصل شده‌اند، **کابل چنداتصالی** (multidrop cable) گفته می‌شود. اترنت با سرعت 2.94 Mbps کار می‌کرد. اترنت یکی از اشکالات عمدهٔ آلوهانت را نیز برطرف کرده بود: هر کامپیوتر قبل از ارسال بستهٔ خود ابتدا به کابل گوش می‌کرد، تا مطمئن شود کس دیگری در همان لحظه در حال استفاده از آن نیست. اگر چنین بود، کامپیوتر تا خالی شدن خط کار خود را عقب می‌انداخت. بدین ترتیب، اترنت توانست با اجتناب از تداخل‌های بی‌مورد به کارایی بالاتری دست یابد. البته آلوهانت امکان انجام چنین کاری را نداشت، چون فرستندهٔ یک جزیره نمی‌توانست امواج فرستنده‌های جزایر دیگر را بشنود.

علیرغم اینکه کامپیوترهای اترنت تا وقتی خط خالی نباشد، اقدام به ارسال نـ ـر کنند، اما مسئلهٔ دیگری ممکنست بروز کند: اگر چند کامپیوتر همـ ـان منتظر خالی شدن خط باشند، و به محض اینکه خط خالی شد، در یک لحظه شروع به ارسال اطلاعات خود کنند، چه خواهد شد؟ راه‌حل این مشکل آن است که هر کامپیوتر در تمام لحظات ارسال اطلاعات خود به خط گوش کند، و اگر متوجه تداخل امواج شد، ابتدا به دیگران اخطار می‌فرستد، و سپس برای مدتی کوتاه (که مقدار آنرا بطور تصادفی تعیین می‌کند) کنار می‌کشد، و بعد از این مدت دوباره سعی

می‌کند؛ اگر در مرتبهٔ بعد باز هم تداخل پیش آمد، مدت انتظار را دو برابر می‌کند، تا بالاخره یکی از آنها فرصت ارسال امواج را پیدا کند.

اترنت زیراکس چنان موفق بود که در سال ۱۹۷۸ شرکتهای DEC ، اینتل و زیراکس استانداردی بنام DIX برای اترنت 10-Mbps وضع کردند. استاندارد DIX در سال ۱۹۸۳ با دو تغییر جزئی به استاندارد IEEE 802.3 تبدیل شد.

متأسفانه زیراکس همواره یکی از شرکتهایی بوده است که اختراعات بسیار مهمی (مانند PC ، اترنت و ماوس) ز آنجا منشأ گرفته، ولی نتوانسته اقدامی برای تجاری کردن آنها بعمل آورد. وقتی زیراکس علاقهٔ چندانی به اترنت نشان نداد (و فقط در استاندارد کردن آن همکاری کرد)، باب متکالف شرکتی بنام 3Com تأسیس کرد تا بتواند برای PC ها کارت شبکهٔ اترنت تولید کند.

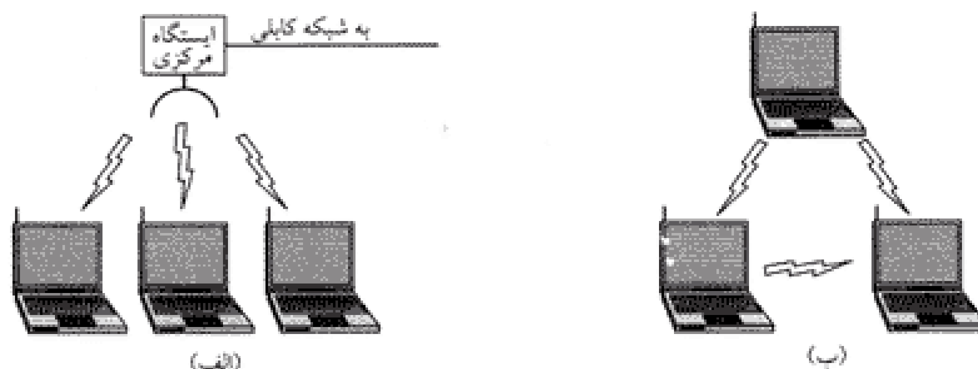
اترنت به رشد و توسعهٔ خود ادامه داد (رشدی که همچنان ادامه دارد)، و امروزه به سرعتهای 100-Mbps و 1000-Mbps دست یافته است – سرعتی که انتظار می‌رود باز هم افزایش یابد. کابل‌کشی، سوئیچینگ و سایر جنبه‌های اترنت نیز بهبود یافته، و ویژگیهای جدیدی به آن اضافه شده است. در فصل ۴ مفصلاً دربارهٔ اترنت صحبت خواهیم کرد.

البته همین جا لازم به توضیح است که اترنت (IEEE 802.3) تنها استاندارد LAN نیست. استانداردهای **خط توکِن** (IEEE 802.4 - Token Bus) و **حلقهٔ توکِن** (IEEE 802.5 - Token Ring) نیز از جمله استانداردهای معروف LAN هستند. وجود سه استاندارد کمابیش ناسازگار برای شبکه‌های محلی بیشتر از اینکه مسئله‌ای فنی باشد، موضوعی سیاسی است. در همان زمان که اترنت در حال استاندارد شدن بود، جنرال موتورز نیز در کار ایجاد شبکه‌ای بود که از همان توپولوژی اترنت (یعنی کابل خطی) استفاده می‌کرد، ولی نوبت ارسال هر کامپیوتر با استفاده از بستهٔ خاصی بنام توکِن، که بین کامپیوترها دست به دست می‌گشت، تعیین می‌شد. هر کامپیوتر فقط زمانی می‌توانست اقدام به ارسال اطلاعات کند که توکِن را در اختیار داشته باشد، و بدین ترتیب مشکل تداخل حل می‌شد. جنرال موتورز اعلام کرد که این تکنیک برای خط تولید کارخانجات اتومبیل‌سازی ضرورت مطلق دارد، و حاضر نبود حتی یک میلیمتر از موضع خود عقب‌نشینی کند. با وجود چنین ادعایی، 802.4 اکنون از صفحهٔ روزگار محو شده است.

غول صنعت کامپیوتر، IBM هم سوگلی خود را داشت: حلقهٔ توکِن. تنها تفاوت این سیستم با شبکهٔ جنرال موتورز آن بود که در اینجا کابل شبکه یک مسیر بسته (حلقه) را تشکیل می‌داد. بر خلاف 802.4 ، 802.5 هنوز در برخی از سایتهای IBM مورد استفاده است (ولی خارج از IBM هیچکس از آن استفاده نمی‌کند). تحقیقاتی در زمینهٔ نسل جدید و پُرسرعت حلقهٔ توکِن با سرعت گیگابیت (802.5v) در جریان است، ولی بنظر نمی‌رسد بتواند با اترنت رقابت کند. خلاصه اینکه، در جنگی که بین اترنت، خط توکِن و حلقهٔ توکِن در گرفت، اترنت پیروز شد، چون اولین و در ضمن از رقبایش بهتر بود.

۱-۵-٤ شبکه‌های محلی بیسیم: 802.11

تقریباً همزمان با به بازار آمدن کامپیوترهای کتابی، بسیاری افراد این رؤیا را در سر می‌پروراندند که بتوانند به محض ورود به جایی که دسترسی اینترنت وجود دارد، بلافاصله و بنحوی جادویی کامپیوترشان به اینترنت متصل شود – و همیشه وقتی رؤیایی وجود دارد، افرادی هم هستند که به فکر محقق کردن آن بیفتند. عملی‌ترین رهیافتی که برای به فعل در آوردن این ایده وجود داشت، مجهز کردن کامپیوترها به فرستنده-گیرنده‌های راد-یو بُرد کوتاه بود – از همین جا بود که شرکتهای متعددی بسرعت شبکه‌های محلی بیسیم را وارد بازار کردند.



شکل ۱-۳۵. شبکهٔ بیسیم (الف) با ایستگاه مرکزی، و (ب) بدون ایستگاه مرکزی.

مشکل اصلی این بود که هیچکدام از این شبکه‌های بیسیم با هم سازگار نبودند، و کامپیوترهایی که به‌بیسیم‌های مختلف مجهز بودند، نمی‌توانستند با هم ارتباط برقرار کنند. بالاخره همه به این نتیجه رسیدند که‌استاندارد کردن شبکه‌های بیسیم می‌تواند ایدهٔ خوبی باشد، و بدنبال آن کمیتهٔ استاندارد IEEE مأمور تدوین این استاندارد شد – استانداردی که 802.11 نام گرفت، و در میان عموم به WiFi معروف است. این یکی از استانداردهای مهم صنعت کامپیوتر است، و شایستهٔ توجه کافی.

استاندارد پیشنهاد شده باید در دو حالت کار می‌کرد:

۱. در شرایط وجود یک ایستگاه مرکزی
۲. در شرایط فقدان ایستگاه مرکزی

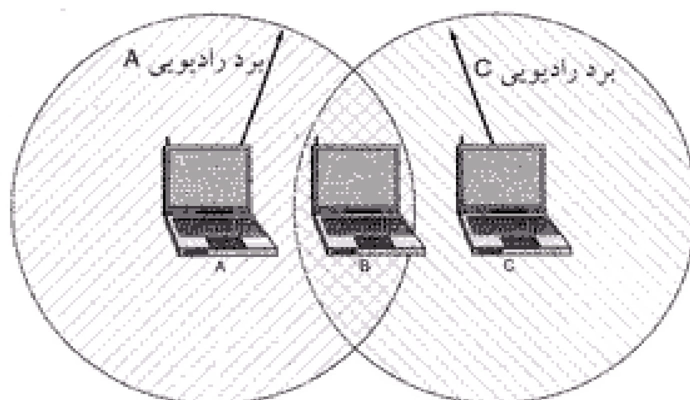
در حالت اول، تمام پیامها باید از طریق ایستگاه مرکزی، که در استاندارد 802.11 به آن **نقطهٔ دسترسی** (access point) گفته می‌شود، مبادله شوند. اما در حالت دوم، کامپیوترها مستقیماً با یکدیگر ارتباط برقرار می‌کنند. این دو حالت را در شکل ۱-۳۵ مشاهده می‌کنید.

شروع می‌شود، و اعداد 1 تا 10 قبلاً استفاده شده بود، نام این استاندارد 802.11 شد. ولی بقیهٔ کار به همین سادگی نبود.

برخی از مهمترین چالشهایی که کمیتهٔ تدوین استاندارد با آنها روبرو بود، عبارت بودند از: انتخاب یک باند فرکانسی مناسب (که ترجیحاً بین‌المللی نیز باشد)؛ محدود بودن بُرد سیگنالهای رادیویی؛ تأمین ایمنی مناسب؛ عمر محدود باتری در کامپیوترهای کتابی؛ مسائل بهداشتی (هنوز این مسئله بدرستی روشن نشده که آیا امواج رادیویی سرطان‌زا هستند یا خیر)؛ پیامدهای متحرک بودن کامپیوترها؛ و بالاخره، ایجاد سیستمی که از نظر پهنای باند ارزش اقتصادی داشته باشد.

با توجه به غالب بودن اترنت در زمان تدوین این استاندارد، کمیته تصمیم گرفت که 802.11 باید در لایه‌های بالاتر از لایهٔ پیوند داده با اترنت سازگار باشد. بویژه، ارسال بسته‌های IP در شبکه‌های بیسیم بایستی دقیقاً به همان روش اترنت باشد.

با این وجود، لایه‌های فیزیکی و لینک داده در این دو سیستم تفاوتهای اساسی با هم دارند. اول اینکه، در اترنت هر کامپیوتر قبل از شروع به ارسال اطلاعات به **اتر** (کابل شبکه) گوش می‌کند، و فقط در صورت خالی بودن آن شروع به ارسال می‌کند. در شبکه‌های بیسیم کار به همین سادگی نیست. برای درک علت آن، به شکل ۱-۳۶ نگاه کنید. فرض کنید کامپیوتر A در حال ارسال اطلاعات به کامپیوتر B است، ولی بُرد امواج آن به کامپیوتر C نمی‌رسد. اگر در این لحظه C بخواهد چیزی به B بفرستد، باید به **اتر** (در اینجا، فضا) گوش کند، ولی آیا عدم



شکل ۱-۳۶. گاهی بُرد امواج رادیویی برای پوشش دادن به تمام شبکه کافی نیست.

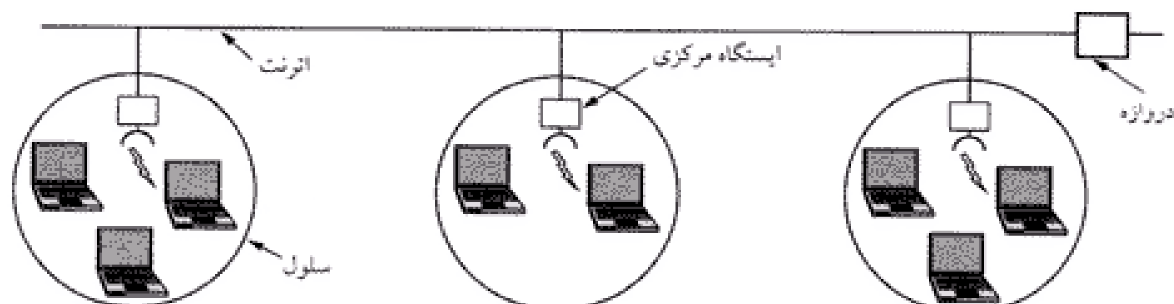
دریافت امواج به معنای آن است که می‌تواند با اطمینان شروع به ارسال کند؛ مسلماً خیر. استاندارد 802.11 باید این مسئله را حل می‌کرد.

مسئلهٔ دومی که باید حل می‌شد این بود که اجسام سخت امواج رادیویی را منعکس می‌کنند، و یک موج می‌تواند چندین بار (و از مسیرهای مختلف) به گیرنده برسد. این تداخل امواج باعث بروز حالتی می‌شود که به آن **محوشدگی چندمسیره** (multipath fading) می‌گویند.

مسئلهٔ سوم این است که بسیاری از نرم‌افزارهای موجود اساساً با چنین وضعیتی (متحرک بودن کامپیوتر) آشنا نیستند. برای مثال، بسیاری از برنامه‌ها دارای لیستی از چاپگرها هستند که می‌توانند روی آنها چاپ کنند. وقتی یکی از این برنامه‌ها (که روی یک کامپیوتر کتابی نصب شده) وارد محیط جدیدی می‌شود، نمی‌تواند تشخیص دهد که فهرست قبلی چاپگرها دیگر در این وضعیت اعتبار ندارد.

مسئلهٔ چهارم این است که وقتی یک کامپیوتر از بُرد یک ایستگاه مرکزی خارج و وارد محدودهٔ ایستگاه دیگری می‌شود، باید مکانیزمی برای این جابجایی وجود داشته باشد. با اینکه تلفنهای همراه دارای چنین مکانیزمی هستند، اما این اتفاق در اترنت نمی‌افتد، و باید بطریقی حل شود. یکی از راه‌حل‌های این مسئله، متصل کردن ایستگاههای مرکزی به یکدیگر از طریق کابل است (شکل ۱-۳۷ را ببینید). از دید دنیای خارج، این شبکه کاملاً شبیه یک شبکهٔ اترنت واحد است. نقطهٔ اتصال سیستم 802.11 با دنیای خارج را **درگاه** (portal) می‌گویند.

بعد از مدتی کار سخت، کمیته موفق به تدوین استانداردی شد که این مسائل (و بسیاری مسائل دیگر) در آن حل شده بود. این شبکهٔ بیسیم با سرعتهای 1-Mbps و 2-Mbps کار می‌کرد. تقریباً بلافاصله، همه لب به شکایت گشودند که این سرعتها بسیار کم است، و کار بر روی استانداردهای سریعتر آغاز شد. در همین زمان کمیتهٔ استاندارد به دو قسمت تقسیم شد، که در سال ۱۹۹۹ هر کدام استاندارد جداگانه‌ای وضع کردند. استاندارد



شکل ۱-۳۷. یک شبکهٔ 802.11 چندسلولی.

802.11a از باند فرکانسی وسیعتری نسبت به 802.11 استفاده می‌کند، و سرعت آن به 54-Mbps می‌رسد. استاندارد 802.11b در همان باند فرکانسی 802.11 کار می‌کند، ولی با استفاده از مدولاسیون متفاوت به سرعت 11-Mbps دست می‌یابد. همانطور که می‌بینید، هر دوی این استانداردها از 802.11 (و حتی از اترنت اولیه) سریعترند، ولی هنوز بدرستی معلوم نیست که کدامیک از آنها برندهٔ نهایی این مسابقه‌اند.

برای اینکه اوضاع از این هم پیچیده‌تر شود، کمیتهٔ 802.11 استاندارد جدیدی بنام 802.11g تدوین کرده، که در باند فرکانسی 802.11b ، ولی با مدولاسیون 802.11a ، کار می‌کند. در فصل ۴ مفصلاً دربارهٔ 802.11 صحبت خواهیم کرد.

شکی نیست که 802.11 انقلاب جدیدی در دنیای کامپیوتر و اینترنت بپا کرده است. فرودگاهها، ایستگاههای قطار، بنادر، هتلها، فروشگاهها، دانشگاهها، مراکز آموزشی (و حتی کافه‌های کوچک) بسرعت در حال نصب شبکه‌های 802.11 هستند. در واقع، 802.11 همان چیزی را به اینترنت داده است، که کامپیوترهای کتابی به دنیای کامپیوتر دادند: تحرک‌پذیری.

۱ـ۶ استانداردهای شبکه

تعداد زیادی سازنده و تأمین‌کنندهٔ قطعات و تجهیزات شبکه وجود دارد، که فکر می‌کنند می‌دانند چگونه باید کار خود را انجام دهند. اما بدون یک عامل هماهنگ‌کننده، این وضعیت می‌تواند به یک آشوب واقعی بینجامد، و در نهایت هیچ کاری هم انجام نشود. تنها راه برای خلاصی از چنین وضعیتی، توافق بر سر استانداردهای شبکه است.

استانداردها نه تنها اجازه می‌دهد تا تجهیزات مختلف بتوانند با هم کار کنند، بلکه فروش محصولات منطبق با استاندارد را نیز افزایش می‌دهند - و فروش بیشتر یعنی تولید انبوه، کاهش هزینه‌ها، طراحی بهتر، کاهش قیمت‌ها، و افزایش مجدد درخواست (و مگر اقتصاد چیزی غیر از این است). در این قسمت نگاهی به استانداردهای بین‌المللی (که از اهمیت زیادی برخوردارند، ولی کمتر شناخته شده‌اند) خواهیم داشت.

استانداردها بر دو نوعند: استانداردهای **بالفعل** (de facto) ، و استانداردهای **قانونی** (de jure) . استانداردهای بالفعل آنهایی هستند که بدون هیچ طرح رسمی بوجود آمده و پذیرفته شده‌اند. کامپیوترهای سازگار با IBM PC (که به اختصار PC خوانده می‌شوند) از جمله استانداردهای بالفعل هستند، چون شرکتهای بسیاری تصمیم گرفتند تا کپی‌های دقیق و کاملی از این نوع کامپیوتر بسازند. یکی دیگر از استانداردهای بالفعل، سیستم عامل **یونیکس** (UNIX) است، که در دانشکده‌های کامپیوتر بعنوان سیستم عامل استاندارد پذیرفته شده است.

از طرف دیگر، استانداردهای قانونی آنهایی هستند که توسط مراجع مسئول بین‌المللی پذیرفته شده‌اند. مراجع بین‌المللی استاندارد به دو دسته تقسیم می‌شوند: آنهایی که طبق معاهدات بین‌المللی تأسیس شده‌اند، و آنهایی که بصورت داوطلبانه شکل گرفته‌اند. در زمینهٔ استانداردهای شبکه‌های کامپیوتری، سازمانهایی از هر دو دسته وجود دارند، که در زیر آنها را معرفی خواهیم کرد.

۱ـ۶ـ۱ مراجع مسئول استانداردهای مخابرات

وضعیت قانونی شرکتهای تلفن از کشوری به کشور دیگر بطرز چشمگیری متفاوت است. در یک سو ایالات متحدهٔ آمریکا قرار دارد، که در آن متجاوز از ۱۵۰۰ شرکت خصوصی در این زمینه مشغول به کار هستند. شرکت AT&T ، قبل از آنکه در سال ۱۹۸۴ طبق قانون ضد تراست به دو بخش تقسیم شود، بزرگترین شرکت خصوصی دنیا بود، و مخابرات این کشور را تحت سلطهٔ کامل خود داشت. این شرکت سرویس تلفن ۸۰ درصد مناطق آمریکا را تأمین می‌کرد، در حالیکه بقیهٔ شرکتها تنها ۲۰ درصد بازار (آن هم اغلب در مناطق روستایی) را در اختیار داشتند. بعد از تقسیم AT&T ، این شرکت اجازه یافت تا فقط در مخابرات راه دور فعالیت کند (آن هم در رقابت با

سایر شرکتها). هفت شرکت منطقه‌ای بل (Bell) ، که از AT&T منشعب و با شرکتهای دیگر متحد شده بودند، نیز خدمات تلفن شهری و تلفن همراه را برعهده گرفتند. بدلیل ادغام‌ها و انشعابات متعدد، صنعت مخابرات در ایالات متحده اکنون از وضعیت ثابتی برخوردار است.

در ایالات متحده، به شرکتهایی که سرویسهای مخابراتی در اختیار عموم قرار می‌دهند، **کاریر عمومی** گفته می‌شود. نوع سرویسها و تعرفهٔ خدمات این شرکتها توسط کمیسیون فدرال مخابرات (برای تماسهای بین‌ایالتی و بین‌المللی) و کمیسیونهای ایالتی (برای تماسهای داخل ایالتی) تعیین می‌شود.

در سوی دیگر، کشورهایی قرار دارند که در آنها تمامی سرویسهای مخابراتی (از جمله، پُست، تلگراف، تلفن، و حتی رادیو و تلویزیون) تحت انحصار مطلق دولت قرار دارد. اکثر کشورهای دنیا هم از این دسته‌اند. در برخی از این کشورها، مخابرات را یک شرکت ملی اداره می‌کند، و در برخی دیگر دولت مستقیماً (از طریق وزارتخانه‌ای بنام **پ.ت.ت**: پُست-تلگراف-تلفن) کارها را در دست دارد. اما، در کل دنیا حرکت یکپارچه‌ای به سمت آزادسازی، رقابت و حذف انحصار دولت آغاز شده است. امروزه اکثر کشورهای اروپایی مخابرات خود را به بخش خصوصی سپرده‌اند، و در دیگر کشورها این فرآیند (هر چند کُند و بطئی) ادامه دارد.

با این همه شرکت و سرویسهای مخابراتی، وجود نوعی استاندارد بین‌المللی برای تضمین ارتباط بین افراد (و کامپیوترها) از کشوری به کشور دیگر ضروری بنظر می‌رسد. البته این نیاز از مدتها قبل آشکار شده بود: در سال ۱۸۶۵، نمایندگانی از کشورهای مختلف اروپایی آنچه را که امروز ITU (**اتحادیهٔ بین‌المللی مخابرات** - International Telecommunication Union) نامیده می‌شود، بنا نهادند. وظیفهٔ این اتحادیه استاندارد کردن مخابرات بین‌المللی (که در آن زمان فقط شامل تلگراف می‌شد) بود. حتی در آن زمان نیز مشخص بود که اگر کلیهٔ کشورها در مورد استفاده از یک کُد یکسان (که همان **کُد مورس** بود) به توافق نرسند، مشکلات عدیده‌ای بروز خواهد کرد. با ورود تلفن به صحنهٔ مخابرات بین‌الملل، ITU وظیفهٔ استاندارد کردن آنرا نیز بر عهده گرفت. در سال ۱۹۴۷، ITU بصورت یکی از سازمانهای تابعهٔ سازمان ملل متحد در آمد.

اتحادیهٔ بین‌المللی مخابرات (ITU) سه بخش عمده دارد:

۱. بخش مخابرات رادیویی (ITU-R)
۲. بخش تدوین استانداردهای مخابراتی (ITU-T)
۳. بخش توسعه (ITU-D)

وظیفهٔ ITU-R تخصیص فرکانسهای رادیویی به متقاضیان در سراسر دنیاست. بخش ITU-T (که بیشتر مد نظر ماست) وظیفهٔ تدوین استاندارد برای سیستمهای مخابراتی (تلفن و داده) را بر عهده دارد. از سال ۱۹۵۳ تا ۱۹۹۳، ITU-T با نام CCITT (که از نام فرانسوی آن - Comité Consultatif International Télégraphique et Téléphonique - گرفته شده بود) شناخته می‌شد. در اول مارس ۱۹۹۳، CCITT برای کاهش بوروکراسی ساختار اداری‌اش را تغییر داد، و نام خود را نیز ITU-T گذاشت. ITU-T و CCITT هر دو توصیه‌هایی را در زمینهٔ مخابرات تلفنی و داده منتشر می‌کردند. امروزه نیز افراد بسیاری به توصیه‌های CCITT مراجعه می‌کنند، اگر چه از سال ۱۹۹۳ این توصیه‌ها نام ITU-T را بر خود دارند.

اعضای ITU-T به چهار دسته تقسیم می‌شوند:

۱. دولت‌ها
۲. اعضای بخش
۳. اعضای وابسته
۴. نمایندگی‌ها

تقریباً تمام کشورهای عضو سازمان ملل متحد (یعنی حدود ۲۰۰ کشور) در ITU-T عضویت دارند. از آنجائیکه ایالات متحدهٔ آمریکا وزارتخانه‌ای بنام «پُست و تلگراف وتلفن» ندارد، فرد دیگری باید نمایندگی آنرا در ITU-T بر عهده بگیرد، که این وظیفه به وزارت امور خارجه محول شده است (شاید به این دلیل که با کشورهای خارجی سروکار دارد). تعداد **اعضای بخش** در ITU-T به حدود ۵۰۰ می‌رسد، که شرکتهای تلفن (مانند AT&T ، ودافون، ورلدکام)، تولیدکنندگان تجهیزات مخابراتی (مانند سیسکو، نوکیا، نورتل)، تولیدکنندگان کامپیوتر (مانند کامپک، سان، توشیبا)، سازندگان چیپ‌های میکروالکترونیک (مانند اینتل، موتورولا، IT)، شرکتهای رسانه‌ای (مانند AOL Time Warner ، CBS ، سونی)، و سایر شرکتهای علاقمند (مانند بوئینگ، سامسونگ، زیراکس) از آن جمله‌اند. تعدادی از سازمانهای علمی غیرانتفاعی و کنسرسیوم‌های صنعتی (مانند IFIP و IATA) نیز در ITU-T عضو بخش هستند. **اعضای وابسته** شرکتهای کوچکتری هستند، که به یکی از مباحث خاص ITU-T علاقمند هستند. **نمایندگی‌ها** نیز آنهایی هستند که بر امور مخابراتی نظارت دارند، مانند کمیسیون مخابرات فدرال ایالات متحده (FCC) .

وظیفهٔ ITU-T ارائهٔ توصیه‌های فنی در زمینهٔ تلفن، تلگراف و مخابرات داده است. این توصیه‌ها اغلب بصورت استانداردهای جهانی پذیرفته می‌شوند، مانند V.24 (در ایالات متحده به نام EIA RS-232 نیز شناخته می‌شود) که نقش و وظیفهٔ پایه‌های رابطهای مودمها و ترمینالهای آسنکرون را مشخص می‌کند.

این نکته را باید تذکر داد که ITU-T فقط توصیه‌های فنی ارائه می‌کند، و دولتها می‌توانند آنها را بپذیرند یا نپذیرند (همانطور که می‌دانید دولتها مثل بچه‌های ده-یازده ساله هستند، یعنی دوست ندارند کسی به آنها دستور بدهد). البته کشوری که استانداردی غیر از استاندارد سایر کشورها را قبول کند، در عمل خود را از سایر کشورهای دنیا ایزوله کرده است - کاری که شاید فقط کرهٔ شمالی مایل به انجام آن باشد. احتمالاً نامگذاری استانداردهای ITU-T بنام «توصیه» فقط کلکی برای آرام کردن ملی‌گرایان کشورهای مختلف است.

کار واقعی ITU-T توسط ۱۴ **گروه مطالعاتی** (که اغلب آنها نزدیک به ۴۰۰ نفر عضو دارند) انجام می‌شود. سرفصل‌هایی که این گروههای مطالعاتی بررسی می‌کنند، از استانداردهای تهیهٔ صورتحساب تلفن گرفته تا سرویسهای چندرسانه‌ای متغیر است. برای اینکه هر گروه بتواند کار خود را بهتر انجام دهد، به چند **دستهٔ کاری** تقسیم می‌شود، که هر یک از این دسته‌ها بنوبهٔ خود به چند **تیم کارشناسی**، و هر تیم تخصصی نیز به **گروههای تخصصی** تقسیم می‌شوند. (مثل اینکه هیچ وقت نمی‌توان از بوروکراسی خلاص شد!)

علیرغم این بوروکراسی، ITU-T کار خود را بخوبی انجام می‌دهد، و از اول تأسیس آن تاکنون نزدیک به ۳۰۰۰ توصیه (که بالغ بر ۶۰,۰۰۰ صفحه می‌شود) تدوین کرده است، که بسیاری از آنها بطور گسترده‌ای مورد استفاده قرار گرفته‌اند. برای مثال، استاندارد V.90 (که دربارهٔ مودمهای 56-kbps است) از توصیه‌های ITU-T می‌باشد.

با رشد و توسعهٔ روزافزون مخابرات راه دور و جهانی شدن آن، استانداردها روز به روز اهمیت بیشتری می‌یابند، و سازمانهای بیشتری مایلند در تدوین آنها شرکت داشته باشند. برای کسب اطلاعات بیشتر دربارهٔ ITU به (Irmer, 1994) مراجعه کنید.

۲-۶-۱ مراجع مسئول استانداردهای بین‌المللی

استانداردهای بین‌المللی توسط **سازمان بین‌المللی استاندارد** (ISO - که البته نام واقعی آن **سازمان بین‌المللی برای تدوین استاندارد**، IOS ، است) تهیه و منتشر می‌شوند. ISO یک سازمان غیرپیمانی داوطلبانه است، در سال ۱۹۴۶ تأسیس شده، و دارای ۸۹ عضو می‌باشد. سازمانهای استاندارد ایالات متحدهٔ آمریکا (ANSI) ، انگلستان (BSI) ، فرانسه (AFNOR) ، آلمان (DIN) و هشتاد وپنج کشور دیگر از اعضای ISO هستند.

تنوع استانداردهایی که ISO منتشر می‌کند، واقعاً جالب و حیرت‌آور است – استانداردهایی وجود دارد که کسی حتی فکر استاندارد بودن آنها را نمی‌کند، مانند استاندارد دانه‌های کاکائو (ISO 2451)، تورهای ماهیگیری (ISO 1530)، لباسهای زیر زنانه (ISO 4416) و غیره. تعداد استانداردهای ISO بالغ بر ۱۳,۰۰۰ است (که استانداردهای OSI نیز از آن جمله‌اند). ISO نزدیک به ۲۰۰ **کمیتهٔ فنی** (TC) دارد، که بترتیب زمان ایجاد شماره‌گذاری شده‌اند و هر کدام در موضوعی خاص تخصص دارند. برای مثال، TC1 با استانداردهای پیچ و مهره سروکار دارد، و TC97 با استانداردهای صنعت کامپیوتر و پردازش اطلاعات. هر TC به چند **زیرکمیته** (SC)، و هر زیرکمیته به چند **گروه کاری** (WG) تقسیم می‌شود.

کار اصلی ISO در WG ها (که متجاوز از ۱۰۰,۰۰۰ عضو داوطلب در سراسر دنیا دارند) انجام می‌شود. بسیاری از این «داوطلبان» در استخدام شرکتهایی هستند که محصول آنها قرار است استاندارد شود؛ برخی دیگر نیز مقامات رسمی کشورهایی هستند که مایلند روش ساخت محصولات کشورشان بصورت استانداردهای جهانی در آید. در برخی از WG ها مقامات متخصصان دانشگاهی نیز حضور دارند.

در زمینهٔ استانداردهای صنعت مخابرات، ISO و ITU-T اغلب با یکدیگر تشریک مساعی دارند (در واقع، ISO یکی از اعضای ITU-T است)، تا از توسعهٔ استانداردهای ناسازگار اجتناب شود.

نمایندهٔ ایالات متحدهٔ آمریکا در ISO **مؤسسهٔ ملی استانداردهای آمریکا** (ANSI) است، که بر خلاف نامش یک مؤسسهٔ غیردولتی و غیرانتفاعی است، و اعضای آن عبارتند از تولیدکنندگان، کاریرهای عمومی و شرکتهای ذینفع. ISO اغلباً استانداردهایی که توسط ANSI وضع می‌شود، را بعنوان استانداردهای بین‌المللی می‌پذیرد. رویه‌ای که در ISO برای پذیرش استانداردها مورد استفاده قرار می‌گیرد، بگونه‌ایست که بیشترین توافق اعضا را بدنبال داشته باشد. این رویه با اعلام نیاز یکی از سازمانهای عضو به یک استاندارد جدید آغاز می‌شود. بدنبال آن یک گروه کاری تشکیل می‌شود، تا برای استاندارد جدید یک **پیش‌نویس کمیته** (CD) تهیه کند. این CD بین سازمانهای عضو توزیع می‌شود، و آنها شش ماه فرصت دارند تا انتقادات خود را مطرح کنند. اگر اکثریت اعضا با این CD موافق باشند، یک سند اصلاح شده بنام **پیش‌نویس استاندارد جهانی** (DIS) برای اظهارنظر و رأی‌گیری منتشر می‌شود. بر اساس نتایج بدست آمده از این دور، متن نهایی بعنوان **استاندارد جهانی** (IS) تنظیم، اصلاح و منتشر می‌شود. در مورد استانداردهایی که اختلاف نظر جدی بین اعضا وجود داشته باشد، امکان دارد CD یا DIS چندین بار اصلاح و به رأی گذاشته شود، تا بتواند رأی کافی بدست آورد، و این فرآیندی است که ممکنست سالها طول بکشد.

در ایالات متحدهٔ آمریکا، **مؤسسهٔ ملی استانداردها و تکنولوژی** (National Institute of Standards and Technology - NIST) که از توابع وزارت بازگانی است، سازمان ملی استاندارد محسوب می‌شود، و تمام خریدهای دولتی (به استثنای خریدهای نظامی، که استانداردهای خاص خود را دارند) بایستی مطابق استانداردهای آن باشد.

یکی دیگر از بازیگران بزرگ در صحنهٔ استانداردهای جهانی، **مؤسسهٔ مهندسان برق و الکترونیک** (IEEE) است، که بزرگترین سازمان حرفه‌ای دنیا محسوب می‌شود. علاوه بر انتشار کتب و مجلات متعدد و برگزاری صدها کنفرانس علمی در سال، IEEE یک گروه تدوین استاندارد نیز دارد که در زمینهٔ مهندسی برق و کامپیوتر فعالیت می‌کند. برای مثال، کمیتهٔ 802 وظیفهٔ تدوین استاندارد شبکه‌های LAN را در IEEE بر عهده دارد، که گروههای کاری آنرا در شکل ۱-۳۸ ملاحظه می‌کنید. میزان موفقیت گروههای کاری 802 چندان بالا نیست، و داشتن عدد 802.x تضمینی برای موفقیت یک استاندارد نیست. البته استثناهایی هم در این زمینه وجود دارد، که استانداردهای 802.3 و 802.11 از آن جمله‌اند.

شماره	سرفصل
802.1	معماری LAN
802.2 ↓	کنترل لینک منطقی
802.3 *	اترنت
802.4 ↓	پاس توکن
802.5	حلقه توکن
802.6 ↓	دوپاس-دو صف
802.7 ↓	گروه مشورتی تکنولوژی های پخشی
802.8 †	گروه مشورتی تکنولوژی های فیبرنوری
802.9 ↓	LAN ایزو سنکرون (برای کاربردهای زمان واقعی)
802.10 ↓	شبکه مجازی و امنیتی
802.11 *	LAN بی سیم
802.12 ↓	تقدم تقاضا (خاص)
802.13	عدد نحس! (کسی آنرا نمی خواهد)
802.14 ↓	مودم کابلی
802.15 *	شبکه های شخصی(بلوتوث)
802.16 *	بی سیم باند وسیع
802.17	حلقه بسته برگشتی

شکل ۱-۳۸. گروههای کاری 802 . گروههای مهم با * مشخص شده‌اند. آنهایی که با ↓ مشخص شده‌اند، به خواب زمستانی رفته‌اند، و گروههایی که با † مشخص شده‌اند، مدتهاست از صفحهٔ روزگار محو شده‌اند.

۱-۶-۳ مراجع مسئول استانداردهای اینترنت

اینترنت نیز مکانیزمهای استانداردسازی خاص خود را دارد، که با ISO و ITU-T بسیار متفاوت است. مهمترین تفاوت آنها اینست که افرادی که در گردهمایی‌های ISO و ITU-T شرکت می‌کنند، کت و شلوار رسمی می‌پوشند، ولی شرکت‌کنندگان در نشست‌های استانداردسازی اینترنت لباس جین به تن می‌کنند (البته اگر تابستان نباشد، و محل اجلاس هم کنار دریا نباشد).

در گردهمایی‌های ISO و ITU-T اغلب نمایندگان شرکتها و دولتها (که استانداردسازی شغل آنهاست) شرکت می‌کنند. آنها به استاندارد بعنوان «یک چیز آسمانی» نگاه می‌کنند، و حاضرند از جانشان برای آن مایه بگذارند. از طرف دیگر، اینترنتی‌ها بی‌نظمی را به عنوان یک اصل پذیرفته‌اند. اما از آنجائیکه بدون حداقلی از اشتراکات اصولاً امکان برقراری هیچ نوع تماسی وجود ندارد، استاندارد را (در کمال تأسف) چیزی لازم می‌دانند.

وقتی آرپانت بوجود آمد، وزارت دفاع ایالات متحدهٔ آمریکا کمیته‌ای رسمی را مأمور نظارت بر آن کرد. در سال ۱۹۸۳، این کمیته به **هیئت نظارت بر فعالیتهای اینترنتی** (Internet Activity Board - IAB) تغییر نام داد، و وظیفهٔ همسو نگه داشتن آرپانت و اینترنت بر عهدهٔ آن گذاشته شد (کاری که می‌توان آنرا به چوپانی یک گله گربه تشبیه کرد!). بعدها نام این کمیته به **هیئت مدیرهٔ معماری اینترنت** (Internet Architecture Board) تغییر کرد. بگونه‌ای که حروف اختصاری آن همچنان IAB باقی ماند.

به هر یک از نزدیک به ده عضو IAB موضوع مهمی برای دنبال کردن محول شد. این اعضا سالی چند بار با هم ملاقات و تبادل نظر می‌کردند، و گزارش کار خود را به NFS و وزارت دفاع (که پشتیبانی مالی آنرا بر عهده داشتند) ارائه می‌کردند. وقتی نیاز به تدوین استاندارد جدیدی حس می‌شد، اعضای IAB آنرا با جار و جنجال زیاد اعلام

می‌کردند، تا دانشجویانی که موتور نرم‌افزاری اینترنت محسوب می‌شدند، بتوانند آنرا پیاده‌سازی کنند. تبادل اطلاعات فنی توسط مقالات و گزارشهایی بنام **نظرخواهی** (Request For Comment - RFC) انجام می‌شد. این RFC ها (که به ترتیب انتشار شماره‌گذاری می‌شوند) در سایت www.ietf.org/rfc نگهداری می‌شوند، تا همهٔ کسانی که مایلند بتوانند به آنها دسترسی داشته باشند. هم اکنون متجاوز از ۳۰۰۰ RFC در این سایت موجود است، که در این کتاب به بسیاری از آنها اشاره خواهیم کرد.

تا سال ۱۹۸۹ اینترنت چنان گسترش یافته بود، که این روش نسبتاً رسمی دیگر نمی‌توانست بکار آید. در آن زمان شرکتهای بسیاری محصولات TCP/IP خود را وارد بازار کرده بودند، و نمی‌خواستند آنها را بصرف اینکه چند پژوهشگر به خیال خود ایده‌های بهتری دارند، عوض کنند. در تابستان ۱۹۸۹، ساختار سازمانی IAB تغییر کرد، و به دو بخش **نیروی پژوهشی اینترنت** (Internet Research Task Force - IRTF) و **نیروی مهندسی اینترنت** (Internet Engineering Task Force - IETF) تقسیم شد، و نیروهای جدیدی (غیر از مجامع تحقیقاتی) وارد آن شدند. در سالهای اول این سازمان‌ها دارای گردش بسته بودند: هر عضو برای دو سال انتخاب می‌شد، و اعضای جدید فقط توسط اعضای قدیمی منسوب می‌شدند. بعدها، **انجمن اینترنت** (Internet Society) بوجود آمد، که تمامی علاقمندان اینترنت می‌توانستند در آن عضویت یابند. انجمن اینترنت از جهاتی شبیه IEEE یا ACM است، و توسط یک هیئت امنای انتخابی، که اعضای IAB را منسوب می‌کند، اداره می‌شود.

هدف از تقسیم IAB آن بود که IRTF بتواند به تحقیقات بلند مدت مشغول شود، در حالیکه IETF به کارهای مهندسی کوتاه مدت اشتغال دارد. IETF خود به چند گروه کاری تقسیم می‌شد، که هر یک روی موضوعی خاص کار می‌کردند. به منظور هماهنگ کردن فعالیتهای این گروهها، مدیران گروههای کاری نشستهای متعددی برگزار می‌کردند. سرفصلهایی که گروههای کاری IETF روی آنها کار می‌کردند، عبارت بودند از: برنامه‌های کاربردی جدید، گردآوری اطلاعات کاربران، یکپارچه‌سازی OSI ، هدایت و آدرس‌دهی، امنیت، مدیریت شبکه، و استانداردسازی. تعداد این گروههای کاری بتدریج افزایش یافت، و اکنون متجاوز از ۷۰ گروه کاری ذیل IETF مشغول به کار هستند.

علاوه بر آن، فرآیند استانداردسازی رسمی‌تری (که شبیه ISO بود) پذیرفته شد. برای آن که یک ایدهٔ جدید به صورت **استاندارد پیشنهادی** (Proposed Standard) در آید، بایستی بطور کامل در یک RFC تشریح شود، و در جامعهٔ اینترنت نیز علاقهٔ کافی نسبت به آن وجود داشته باشد. سپس، برای آنکه این استاندارد به مرحلهٔ **پیش‌نویس استاندارد** (Draft Standard) ارتقاء یابد، بایستی بصورت واقعی پیاده‌سازی شده و به مدت حداقل ۴ ماه در ۲ سایت اینترنتی بطور همه جانبه تست شود. اگر IAB متقاعد شود که ایدهٔ اصلی خوب است و نرم‌افزار نیز بخوبی کار می‌کند، می‌تواند این RFC را رسماً بعنوان یک استاندارد اینترنتی اعلام کند. برخی از این استانداردها حتی بصورت استانداردهای نظامی (MIL-STD) در آمده‌اند، و رعایت آنها در محصولاتی که به وزارت دفاع فروخته می‌شوند، اجباریست. دیوید کلارک نقل قول جالبی دربارهٔ استانداردهای اینترنت دارد که اکنون بسیار معروف است: «اجماع نظری ناقص، بهمراه کُدی که کار می‌کند.»

۷. واحدهای اندازه‌گیری

برای اجتناب از هر گونه سوء‌تفاهم، لازم است تأکید کنیم که در این کتاب بجای واحدهای اندازه‌گیری انگلیسی (با آن اندازه‌ها و ضرایب عجیب و غریب) از سیستم متریک استفاده شده است. در شکل ۱-۳۹ پیشوندهای اصلی سیستم متریک را مشاهده می‌کنید. اغلب از حروف اختصاری این پیشوندها استفاده خواهیم کرد، و واحد اعداد بزرگتر از 1 با حروف بزرگ می‌نویسیم، مانند KB و MB (البته با یک استثناء تاریخی: حروف اختصاری

پیشوند	عدداعشاری	توان	پیشوند	عدداعشاری	توان
Kilo	1,000	10^{3}	milli	0.001	10^{-3}
Mega	1,000,000	10^{6}	micro	0.000001	10^{-6}
Giga	1,000,000,000	10^{9}	nano	0.000000001	10^{-9}
Tera	1,000,000,000,000	10^{12}	pico	0.000000000001	10^{-12}
Peta	1,000,000,000,000,000	10^{15}	femto	0.000000000000001	10^{-15}
Exa	1,000,000,000,000,000,000	10^{18}	atto	0.000000000000000001	10^{-18}
Zetta	1,000,000,000,000,000,000,000	10^{21}	zepto	0.000000000000000000001	10^{-21}
Yotta	,000,000,000,000,000,000,000	10^{24}	yocto	0.000000000000000000000001	10^{-24}

شکل ۱-۳۹. پیشوندهای اصلی سیستم متریک.

کیلوبیت/ثانیه بصورت kbps نوشته می‌شود). برای مثال، یک خط انتقال 1-Mbps در هر ثانیه 10^6 بیت اطلاعات را منتقل می‌کند، و 100 psec معادل 10^{-10} ثانیه است. برای تمایز بین پیشوندهای «میلی» و «میکرو» - که هر دو با m شروع می‌شوند -، اولی را با "m" و دومی را با "μ" نمایش می‌دهیم.

همچنین اشاره به این نکته خالی از فایده نیست که، واحدهایی که در صنعت کامپیوتر برای اندازه‌گیری ظرفیت حافظه، دیسک، و فایل بکار برده می‌شوند، کمی با سایر واحدها تفاوت دارند. در اینجا، «کیلو» بجای 10^3 (معادل 1000) معنای 2^{10} (یا 1024) می‌دهد، چون ظرفیت حافظه همیشه توانی از ۲ است - بنابراین، وقتی گفته می‌شود 1-KB ، بمعنای ۱۰۲۴ بایت است، نه ۱۰۰۰ بایت. به همین ترتیب، 1-MB معادل 2^{20} (1,048,576 بایت) است، 1-GB معادل 2^{30} (1,073,741,824 بایت)، و 1-TB معادل 2^{40} (1,099,511,627,776 بایت). از طرف دیگر، در صحبت از سرعت انتقال داده‌ها، دیگر اعداد توانی از ۲ نیستند؛ برای مثال، خط 1-kbps دقیقاً 1000 بیت/ثانیه داده منتقل می‌کند، و یک شبکهٔ 1-Mbps دقیقاً با سرعت 1,000,000 بیت/ثانیه کار می‌کند. متأسفانه، بسیاری از افراد قادر به تفکیک این دو نیستند (بویژه در مورد ظرفیت دیسکها). برای اجتناب از هر گونه ابهام، در این کتاب واحدهای KB ، MB ، و GB بترتیب معادل 2^{10} ، 2^{20} ، و 2^{30} بایت، و واحدهای kbps ، Mbps ، و Gbps بترتیب معادل 10^3 ، 10^6 ، و 10^9 بیت/ثانیه در نظر گرفته خواهند شد.

۱-۸ طرح کلی مباحث کتاب

در این کتاب شبکه‌های کامپیوتری از جنبهٔ نظری و عملی مورد بررسی قرار گرفته‌اند. اکثر فصول کتاب با بحث در کلیات موضوع موردنظر شروع شده، و با بررسی مثالها و نمونه‌های عملی ادامه می‌یابد. این نمونه‌ها از اینترنت و شبکه‌های بیسیم انتخاب شده‌اند، چون در عین اهمیت بسیار با یکدیگر تفاوتهای اساسی نیز دارند. هر گاه لازم بوده به مثالهای دیگر نیز استناد کرده‌ایم.

در این کتاب کار خود را بر اساس مدل ترکیبی شکل ۱-۲۴ بنا نهاده‌ایم، و از فصل آینده بحث دربارهٔ سلسله مراتب پروتکلهای این مدل را (از پائین به بالا) شروع خواهیم کرد. در فصل ۲ ابتدا پیش‌زمینه‌ای دربارهٔ سیستمهای مخابرات داده (که شامل سیستمهای کابلی، بیسیم و ماهواره‌ای می‌شود) بدست خواهیم داد. این سیستمها که به لایهٔ فیزیکی مربوط می‌شوند، بیشتر از جنبهٔ معماری مورد بحث قرار گرفته‌اند تا جنبهٔ سخت‌افزاری. در این فصل نمونه‌های متعددی را مورد بررسی قرار داده‌ایم، از شبکه‌های سوئیچینگ تلفن معمولی و تلفن همراه گرفته، تا

شبکه‌های تلویزیون کابلی.

در فصل ۳ لایهٔ پیوند داده و پروتکلهای آن (بهمراه تحلیل این پروتکلها) به کمک مثالهای متعدد مورد بحث قرار گرفته است. پس از آن، تعدادی از پروتکلهای مهم این لایه، از جمله HDLC (که در شبکه‌های با سرعت کم تا متوسط مورد استفاده قرار می‌گیرد) و PPP (که در اینترنت کاربرد دارد) را مورد بررسی قرار خواهیم داد.

فصل ۴ با زیرلایهٔ دسترسی داده (که بخشی از لایهٔ پیوند داده است) سروکار دارد. بحث اصلی در اینجا نحوهٔ دسترسی به شبکه در جاهائیست که کانالهای فیزیکی شبکه به صورت اشتراکی مورد استفاده قرار می‌گیرند (مانند شبکه‌های LAN و برخی از شبکه‌های ماهواره‌ای). در این فصل نیز نمونه‌های متعددی (از جمله اترنت، LAN بیسیم، MAN بیسیم، بلوتوث، و شبکه‌های ماهواره‌ای) مورد بررسی قرار گرفته است. دربارهٔ **پل** (bridge) و **سوئیچ** (switch) نیز در همین فصل صحبت کرده‌ایم.

فصل ۵ به لایهٔ شبکه، بویژه **مسیریابی** (routing) و الگوریتمهای آن (استاتیک و دینامیک)، اختصاص دارد. حتی با بهترین الگوریتمهای مسیریابی، اگر بار بیش از حد به یک شبکه تحمیل شود، امکان بروز **ازدحام** (congestion) در آن وجود دارد، بنابراین یکی از مباحث مهم این فصل ازدحام و راههای جلوگیری از آن (و از آن هم فراتر، تضمین نوعی کیفیت سرویس) است. نحوهٔ اتصال شبکه‌های غیرمتجانس و رفع مشکلات آن از دیگر مباحث این فصل است. لایهٔ شبکه از اهمیت زیادی در اینترنت برخوردار است.

در فصل ۶ دربارهٔ لایهٔ انتقال (و بویژه پروتکلهای اتصال-گرا) مفصلاً صحبت خواهیم کرد. حتی یکی از سرویسهای سادهٔ این لایه بصورت عملی (بهمراه کُد آن) ارائه شده است، تا با نحوهٔ پیاده‌سازی آن بیشتر آشنا شوید. پروتکلهای اینترنتی TCP و UDP (و کارایی آنها) نیز مفصلاً در این فصل مورد بررسی قرار گرفته‌اند. از دیگر مباحثی که در فصل ۶ مطرح خواهد شد، شبکه‌های بیسیم است.

فصل ۷ با لایهٔ کاربرد (و پروتکلها و برنامه‌های آن) سروکار دارد. اولین مبحث این فصل، دفترچه تلفن اینترنت یعنی DNS است. ایمیل و پروتکلهای آن مبحث بعدی فصل ۷ است. پس از آن سراغ وب می‌رویم، و مباحثی مانند محتویات استاتیک و دینامیک، فعل و انفعالات سمت مشتری و سرویس‌دهنده، پروتکلهای آن، کارایی وب، و وب بیسیم را مورد بررسی قرار خواهیم داد. در پایان نیز، دربارهٔ شبکه‌های چندرسانه‌ای (صدا و تصویر **جویباری** – streaming –، و رادیوی اینترنتی) صحبت خواهیم کرد.

فصل ۸ دربارهٔ امنیت شبکه است. از آنجائیکه مباحث این فصل با تمام لایه‌ها سروکار دارد، آنرا در آخر (بعد از پایان بحث لایه‌های شبکه) آورده‌ایم. این فصل را با معرفی تکنولوژیهای **رمزنگاری** (cryptography) آغاز کرده‌ایم، و نشان داده‌ایم که چگونه می‌توان به کمک این تکنولوژیها امنیت مخابرات داده، ایمیل و وب را تأمین کرد. این فصل را با بحثی دربارهٔ تقابل بین امنیت و حریم خصوصی افراد، آزادی بیان، سانسور (و سایر موضوعاتی که با امنیت شاخ به شاخ می‌شوند) به پایان رسانده‌ایم.

در فصل ۹ مقالات و کتابهایی را که می‌توانید برای مطالعهٔ بیشتر دربارهٔ موضوعات هر فصل به آنها مراجعه کنید، به همان ترتیب فصول کتاب آورده‌ایم. در این فصل یک کتابنامهٔ مفصل در زمینهٔ موضوعات مطرح شده در کتاب نیز آورده شده است.

در سایت وب مؤلف کتاب

http://www.prenhall.com/tanenbaum

می‌توانید لینکهای متعددی به درسنامه‌ها، صفحات پرسش و پاسخ، شرکتها، کنسرسیومهای صنعتی، سازمانهای حرفه‌ای، سازمانهای استاندارد، تکنولوژیها، و مقالات تخصصی پیدا کنید.

۹-۱ خلاصه

از شبکه‌های کامپیوتری می‌توان برای مقاصد مختلفی (در شرکتها، یا برای افراد عادی) استفاده کرد. در شرکتها، شبکه می‌تواند دسترسی به منابع اطلاعاتی را برای تمام کارکنان فراهم آورد. در این شبکه‌ها معمولاً از مدل مشتری-سرویس‌دهنده (که در آن منابع مشترک روی کامپیوترهای قدرتمندی موسوم به **سرویس‌دهنده** - server - قرار می‌گیرند) استفاده می‌شود. شبکه برای افراد عادی امکان دسترسی به منابع اطلاعاتی یا تفریحی را فراهم می‌آورد. امروزه افراد بسیاری با یک مودم از منزل خود به شرکتهای خدمات اینترنتی (ISP) متصل شده، و از امکانات آن استفاده می‌کنند. یکی از زمینه‌هایی که امروزه بسرعت رو به گسترش است، شبکه‌های بیسیم است، که امکان دسترسی به اینترنت را حتی از تلفنهای همراه به افراد می‌دهد.

در یک تقسیم‌بندی بسیار کلی، شبکه‌ها را می‌توان به شبکه‌های محلی (LAN)، شبکه‌های شهری (MAN)، شبکه‌های گسترده (WAN) و شبکهٔ‌شبکه‌ها (internetwork) تقسیم کرد، که هر کدام برای خود دارای ویژگیها، سرعت، تکنولوژی و جایگاه خاصی می‌باشد. شبکهٔ محلی (LAN) سرعت بالایی دارد، ولی معمولاً به یک ساختمان منفرد محدود می‌شود. محدودهٔ جغرافیایی شبکهٔ شهری (MAN) - همانطور که از نام آن برمی‌آید - یک شهر است؛ تلویزیون کابلی نمونه‌ای از شبکه‌های شهری است. شبکه‌های گسترده (WAN) یک کشور یا قاره را در بر می‌گیرند. شبکه‌های LAN و MAN سوئیچ‌نشده هستند (یعنی، **مسیریاب** - router - ندارند)، در حالیکه شبکه‌های WAN سوئیچ‌شده اند. امروزه شبکه‌های بیسیم (بویژه LAN بیسیم) از محبوبیت روزافزونی برخوردارند. از اتصال چند شبکه به یکدیگر نیز یک شبکهٔ‌شبکه‌ها شکل می‌گیرد.

نرم‌افزار شبکه از پروتکلها (که قواعد حاکم بر ارتباط پروسس‌ها را تعیین می‌کنند) تشکیل می‌شود. پروتکلها یا **اتصال-گرا** (connection-oriented) هستند یا **غیرمتصل** (connectionless). اغلب شبکه‌ها از مدل سلسله مراتبی پروتکلها استفاده می‌کنند، که در آن هر لایه سرویسهایی را در اختیار لایه‌های بالاتر قرار می‌دهد، و آنرا از جزئیات کار در لایه‌های پائینتر ایزوله می‌کند. **مجموعه پروتکل** (protocol stack) های مهم امروزی عمدتاً بر مبنای دو مدل OSI و TCP/IP بنا شده‌اند. هر دوی این مدلها دارای لایه‌های شبکه، انتقال و کاربرد هستند، ولی در لایه‌های دیگر با هم فرق دارند. هنگام طراحی پروتکلها باید به مسائلی از قبیل **مالتی‌پلکس کردن** (multiplexing)، **کنترل جریان** (flow control)، **کنترل خطا** (error control) و مانند آنها توجه ویژه مبذول داشت. (بخش عمده‌ای از این کتاب به پروتکلها و طراحی آنها اختصاص دارد.)

وظیفهٔ شبکه ارائهٔ سرویس به کاربران است، که این سرویسها نیز می‌توانند **اتصال-گرا** یا **غیرمتصل** باشند. در برخی شبکه‌ها، یک لایه سرویس غیرمتصل در اختیار می‌گذارد، در حالیکه لایهٔ بالاتر سرویس اتصال-گرا ارائه می‌کند.

معروفترین شبکه‌های موجود عبارتند از: اینترنت، ATM، اترنت و LAN بیسیم (IEEE 802.11). اینترنت محصول تکامل شبکهٔ **آرپانت** (ARPANET - شبکهٔ سوئیچ بستهٔ وزارت دفاع ایالات متحدهٔ آمریکا) بود. اینترنت در واقع امروزه شبکه‌ایست از هزاران شبکهٔ دیگر، نه یک شبکهٔ واحد. مشخصهٔ اصلی اینترنت استفاده از مجموعه پروتکل TCP/IP است. شبکه‌های ATM بیشتر در سیستمهای تلفن، و برای مخابرات راه دور، بکار می‌روند. **اترنت** (Ethernet) نیز محبوبترین تکنولوژی LAN است، که در اغلب شرکتها و دانشگاهها از آن استفاده می‌شود. و بالاخره، شبکه‌های LAN بیسیم که (با سرعت خیره‌کنندهٔ 54 Mbps) بسرعت در حال گسترش هستند.

برای آنکه چند کامپیوتر بتوانند با هم ارتباط برقرار کنند، مهمترین موضوع وجود استانداردهای متعدد (سخت‌افزاری و نرم‌افزاری) است. وظیفهٔ تدوین استاندارد برای شبکه‌های کامپیوتری بر عهدهٔ سازمانهایی از قبیل ITU-T، ISO، IEEE و IAB گذاشته شده است.

مسائل

۱. فرض کنید سگی از نژاد سنت‌برنارد (نوعی سگ قوی‌هیکل که در کشور کوهستانی سوئیس برای نجات کوهنوردان سانحه‌دیده تربیت می‌شود) بنام بِرنی دارید، و او را برای حمل جعبه‌ای حاوی سه نوار 8mm آموزش داده‌اید (به هر حل پُر شدن دیسک هم نوعی موقعیت اورژانس است). هر نوار 7 GB ظرفیت دارد، و برنی می‌تواند تحت هر شرایطی با سرعت 18 km/h حرکت کند. تا چه فاصله‌ای نرخ انتقال اطلاعات برنی همچنان از خط انتقالی با سرعت 150 Mbps (بدون در نظر گرفتن سرآیند) بیشتر است؟

۲. یکی از گزینه‌های رقیب شبکه‌های LAN سیستمهای تسهیم زمانی (time sharing) بزرگ (بهمراه ترمینالهایی که کاربران از آنها استفاده می‌کنند) است. دو مزیت شبکه‌های LAN مبتنی بر مدل مشتری-سرویس‌دهنده را نسبت به سیستمهای فوق بیان کنید.

۳. کارایی سیستمهای مشتری-سرویس‌دهنده به دو ویژگی مهم شبکه وابسته است: پهنای باند (bandwidth - حداکثر تعداد بیت‌هایی که شبکه می‌تواند در هر ثانیه منتقل کند)، و زمان تأخیر (latency time - مدت زمانی که طول می‌کشد تا اولین بیت از مشتری به سرویس‌دهنده - یا بالعکس - برسد). دو مثال از شبکه‌ای با پهنای باند زیاد و زمان تأخیر زیاد، و پهنای باند کم و زمان تأخیر کم بزنید.

٤. غیر از پهنای باند زیاد و زمان تأخیر کم، چه شرایط دیگری باید فراهم باشد تا یک شبکه کیفیت مناسبی برای سرویس صدای دیجیتال داشته باشد؟

۵. یک از عوامل مهم در میزان تأخیر سیستمهای سوئیچینگ بستهٔ مبتنی بر مدل ذخیره-هدایت، مدت زمانیست که صرف ذخیره و هدایت بسته در یک سوئیچ می‌شود. آیا زمان سوئیچینگ 10 μsec برای یک سیستم مشتری-سرویس‌دهنده که بین کالیفرنیا و نیویورک کار می‌کند، عامل مهمی محسوب می‌شود. سرعت انتشار امواج الکترومغناطیس در سیم مسی را 0.667 سرعت نور در نظر بگیرید.

۶. یک سیستم مشتری-سرویس‌دهنده از شبکهٔ ماهواره‌ای، که ماهوارهٔ آن در ارتفاع 40,000 km سطح زمین قرار گرفته، استفاده می‌کند. زمان تأخیر پاسخ در بهترین حالت چقدر است؟

۷. در آینده‌ای نه چندان دور مردم می‌توانند مستقیماً از طریق ترمینالهای متصل به شبکه‌های کامپیوتری به لوایح و طرحهای قانونی رأی بدهند، و دیگر نیازی به مجالس قانونگذاری وجود نخواهد داشت. جنبه‌های مثبت چنین دموکراسی مستقیمی نسبتاً روشن است؛ کمی دربارهٔ نقاط منفی آن بحث کنید.

۸. می‌خواهیم پنج مسیریاب را در یک زیرشبکهٔ نقطه-به-نقطه به هم وصل کنیم. هر زوج از این مسیریاب‌ها را می‌توان با یک خط پُرسرعت، خطی با سرعت متوسط، و یا یک خط کم سرعت به هم متصل کرد، و یا اینکه اصلاً آنها را به هم وصل نکرد. اگر یک کامپیوتر بتواند هر توپولوژی را در 100 ms طراحی و بررسی کند، چه مدت طول می‌کشد تا تمام توپولوژیهای ممکن را بررسی کند؟

۹. تعداد $2^n - 1$ مسیریاب در یک درخت باینری متقارن (یک مسیریاب در هر گره) به هم متصل شده‌اند. مسیریاب i برای ارتباط با مسیریاب j، باید ابتدا پیام خود را به ریشهٔ این درخت بفرستد، تا از آنجا بدست مسیریاب j برسد. (با فرض اینکه تمام مسیریاب یکسان هستند) عبارتی بنویسید که تعداد متوسط پرشهای لازم برای رسیدن پیام یک مسیریاب به مسیریاب دیگر را (برای n های بزرگ) بدست دهد.

۱۰. یکی از نقاط منفی زیرشبکه‌های پخشی ظرفیتی است که در اثر اقدام به پخش همزمان توسط چند کامپیوتر از دست می‌رود (تصادم - collision). اجازه دهید مسئله را ساده کرده، و فرض کنیم زمان به برشهای مساوی تقسیم شده، و در هر برش زمانی n کامپیوتر با احتمال p اقدام به پخش روی شبکه می‌کنند. چه

کسری از برشهای زمانی در اثر بروز حالت تصادم تلف خواهند شد؟

۱۱. دو دلیل برای استفاده از پروتکلهای لایه‌ای ارائه کنید.

۱۲. رئیس یک شرکت رنگسازی ایدهٔ جدیدی برای تولید یک محصول بدیع و نوظهور دارد. موضوع را با ادارهٔ حقوقی شرکت در میان می‌گذارد، و آنها هم از ادارهٔ مهندسی درخواست کمک می‌کنند. سرپرست ادارهٔ مهندسی دربارهٔ زوایای مختلف این طرح با یکی از همتایان خود در شرکتی دیگر مشورت می‌کند، که وی نیز موضوع را به ادارهٔ حقوقی شرکت خود منتقل می‌کند. در پایان نیز، رؤسای دو شرکت بر سر جزئیات مالی معامله با یکدیگر به مذاکره می‌پردازند. آیا می‌توان این سناریو را مطابق با پروتکلهای چند لایهٔ مدل OSI دانست؟

۱۳. دو تفاوت عمده بین ارتباطات اتصال-گرا و غیرمتصل چیست؟

۱۴. دو شبکه سرویسهای اتصال-گرای قابل‌اعتماد ارائه می‌کنند: یکی از آنها بصورت استریم بایت، و دیگری بصورت استریم پیام. آیا این دو یکسان هستند؟ اگر پاسخ مثبت است، چه دلیلی برای نامگذاری جداگانهٔ آنها وجود دارد؟ اگر خیر، تفاوت آنها در چیست؟

۱۵. در بحث پروتکلهای شبکه، «**مذاکره**» (negotiation) چه معنایی دارد؟ یک مثال بزنید.

۱۶. در شکل ۱-۱۹ یک سرویس نشان داده شده است. آیا سرویس دیگری در این شکل وجود دارد؟ اگر آری، کجا؟ اگر خیر، چرا؟

۱۷. در برخی از شبکه‌ها، لایهٔ پیوند داده با درخواست ارسال مجدد فریمهایی که بدرستی دریافت نشده‌اند، نوعی کنترل خطا انجام می‌دهد. اگر احتمال خراب شدن یک فریم p باشد، متوسط تعداد دفعاتی که یک فریم باید فرستاده شود، چقدر است؟ فرض کنید فریمهای **تصدیق دریافت** (acknowledgement) هرگز خراب نمی‌شوند.

۱۸. کدامیک از لایه‌های OSI وظایف ذیل را بر عهده دارند:
(الف) تقسیم استریم بیتها به فریم
(ب) تعیین مسیری در زیرشبکه، که باید از آن استفاده شود

۱۹. اگر واحد تبادل داده در لایهٔ پیوند داده **فریم**، و در لایهٔ انتقال **بسته** نام داشته باشد، فریمها در بسته پیچیده می‌شوند، یا بسته‌ها در فریم؟ توضیح دهید.

۲۰. ساختار سلسله مراتبی پروتکلهای یک سیستم n لایه دارد. برنامه‌های کاربردی در این سیستم پیامهایی بطول M بایت تولید می‌کنند، و در هر لایه یک سرآیند h بایتی به پیام لایهٔ بالاتر اضافه می‌شود. چه کسری از پهنای باند شبکه را این سرآیندها اشغال می‌کنند؟

۲۱. دو شباهت و دو تفاوت مدلهای مرجع OSI و TCP/IP را بیان کنید.

۲۲. تفاوت اصلی TCP و UDP چیست؟

۲۳. زیرشبکهٔ شکل ۱-۲۵(ب) طوری طراحی شده که در مقابل حملات هسته‌ای دوام بیاورد. چند بمب اتمی لازم است تا این زیرشبکه به دو قسمت کاملاً مجزا تقسیم شود؟ فرض کنید برای نابود کردن هر گره و تمام لینکهای متصل به آن یک بمب اتمی کافیست.

۲۴. تخمین زده شده است که اینترنت هر ۱۸ ماه دو برابر می‌شود. (و با اینکه کسی واقعاً تعداد آنها را نمی‌داند) طبق برآوردهای تخمینی در سال ۲۰۰۱ تعداد کامپیوترهای اینترنت ۱۰۰ میلیون بوده است. با استفاده از این

مفروضات، تعداد کامپیوترهای اینترنت در سال ۲۰۱۰ را محاسبه کنید. آیا این عدد را باور می‌کنید؟ توضیح دهید.

۲۵. هنگام تبادل فایل بین دو کامپیوتر، دو استراتژی تصدیق دریافت ممکن است. در روش اول، فایل به قطعاتی تقسیم شده، و برای هر قطعه تصدیق دریافت جداگانه مطالبه می‌شود، اما برای کل فایل، خیر. در روش دوم، برای تک تک قطعات تصدیق دریافت مطالبه نمی‌شود، ولی در پایان دریافت کل فایل باید به تأئید طرف مقابل برسد. دربارهٔ این دو روش بحث کنید.

۲۶. چرا ATM از سلولهای کوچک و با اندازه ثابت استفاده می‌کند؟

۲۷. هر بیت در استاندارد اولیهٔ 802.3 چند متر طول داشت؟ سرعت کار این شبکه را 10 Mbps ، و سرعت سیگنالهای الکتریکی در کابل کواکسیال را 2/3 سرعت نور در خلاء فرض کنید.

۲۸. ابعاد یک تصویر 768 × 1024 پیکسل است، و هر پیکسل آن 3 بایت جا می‌گیرد (و فرض کنید این تصویر فشرده نشده است). انتقال این تصویر روی یک خط 56-kbps چقدر طول می‌کشد؟ روی یک خط 1-Mbps چقدر؟ روی شبکهٔ اترنت 10-Mbps چقدر؟ و روی شبکهٔ اترنت 100-Mbps چقدر؟

۲۹. اترنت و شبکه‌های بیسیم شباهتها و تفاوتهایی دارند. یکی از ویژگیهای اترنت اینست که در هر لحظه فقط یک فریم می‌تواند روی شبکه منتقل شود. آیا در شبکه‌های 802.11 نیز چنین است؟ توضیح دهید.

۳۰. نصب شبکه‌های بیسیم بسیار ساده است، که همین باعث ارزانی آنها شده است، چون معمولاً هزینه‌های نصب در مقابل هزینهٔ تجهیزات رقم قابل توجهی را تشکیل می‌دهد. با این حال، این نوع شبکه معایبی نیز دارد. دو تا از این معایب را نام ببرید.

۳۱. دو مزیت و دو عیب برای استاندارد کردن پروتکلهای شبکه بشمارید.

۳۲. وقتی یک سیستم از دو قسمت ثابت و متحرک تشکیل می‌شود (مانند درایو CD ، و دیسک CD-ROM)،استاندارد کردن آن از اهمیت زیادی برخوردار است، تا محصولات شرکتهای مختلفی که این قطعات را تولید می‌کنند، با هم سازگار باشد. سه مثال از خارج صنعت کامپیوتر بزنید، که چنین استانداردی در آنها وجود دارد. حال سه مثال از خارج صنعت کامپیوتر بزنید، که چنین استانداردی در آنها وجود نداشته باشد.

۳۳. تمام فعالیتهای خود را که در طول شبانه روز بنحوی با شبکه سروکار دارد، فهرست‌وار بنویسید. اگر این شبکه‌ها به یکباره از کار بیفتند، چه تأثیری روی زندگی شما خواهد گذاشت؟

۳٤. شبکه‌هایی راکه در محل کار یا تحصیل خود می‌شناسید (بهمراه نوع، توپولوژی، و روش سوئیچینگ آنها)، مشخص کنید.

۳۵. برنامهٔ ping به شما اجازه می‌دهد تا بسته‌ای را به یک مقصد مشخص فرستاده، و زمان رفت و برگشت آنرا محاسبه کنید. از این برنامه برای تعیین زمان رفت و برگشت بسته‌ها به چند نقطهٔ مختلف استفاده کنید. با استفاده از این اطلاعات، نمودار زمان انتقال بسته‌ها روی اینترنت (در جهت رفت) را بر حسب فاصله رسم کنید. بهتر است برای این منظور از دانشگاههای مهم و سرشناس (که محل آنها دقیقاً مشخص است) استفاده کنید. برای مثال، سایت berkeley.edu در برکلی-کالیفرنیا است، سایت mit.edu در کمبریج-ماساچوست،

سایت vu.nl در آمستردام-هلند، سایت www.usyd.edu.au در سیدنی-استرالیا، و سایت www.uct.ac.za در کیپ‌تاون-افریقای جنوبی.

۳۶. سری به سایت IETF (به آدرس www.ietf.org) بزنید، و ببینید چکار می‌کنند. یکی از پروژه‌هایی را که به آن علاقه دارید، انتخاب کرده، و دربارهٔ آن مقاله‌ای نیم صفحه‌ای بنویسید.

۳۷. استانداردسازی در دنیای شبکه‌های کامپیوتری بسیار مهم است، و همانطور که دیدید سازمانهای ITU و ISO این وظیفه را بر عهده دارند. به سایت وب این سازمانها (www.iyu.org و www.iso.org) رفته، و با کارهای استانداردسازی آنها آشنا شوید. گزارش کوتاهی دربارهٔ انواع چیزهایی که این دو سازمان استاندارد کرده‌اند، بنویسید.

۳۸. اینترنت از تعداد زیادی شبکه‌های مختلف تشکیل شده است، که آرایش آنها توپولوژی اینترنت را مشخص می‌کند. در خود اینترنت اطلاعات زیادی در زمینهٔ توپولوژی آن وجود دارد. با استفاده از یک **موتور جستجو** (search engine) در باره این موضوع تحقیق کرده، و خلاصه‌ای از یافته‌های خود را در یک گزارش بنویسید.



لایه فیزیکی

در این فصل پائین‌ترین لایهٔ سلسله مراتب شکل ۱-۲۴ را مورد بررسی قرار خواهیم داد. مشخصات مکانیکی، الکتریکی و تایمینگ (همزمانی) شبکه در این لایه تعریف می‌شود. برای شروع کمی دربارهٔ تئوری مخابرات صحبت می‌کنیم، فقط برای اینکه نشان دهیم دست طبیعت چه محدودیتهایی را در زمینهٔ انتقال داده‌ها به ما تحمیل کرده است.

سپس با رسانه‌های فیزیکی که برای انتقال داده‌ها از آنها استفاده می‌شود، آشنا می‌شوید: رسانه‌های هدایت‌پذیر (مانند سیم مسی و فیبر نوری)، بیسیم (امواج رادیویی زمینی)، و ماهواره. آشنایی با این رسانه‌ها برای شناخت تکنولوژیهای مدرن مخابراتی اهمیت اساسی دارد.

در ادامه سه نمونه از سیستمهای مخابراتی که در شبکه‌های کامپیوتری کاربرد گسترده‌ای دارند، را مفصلاً مورد بررسی قرار خواهیم داد: سیستم تلفن ثابت، شبکهٔ تلفن همراه، و تلویزیون کابلی. در تمام این سیستمها از فیبر نوری بعنوان **ستون فقرات** (backbone) استفاده می‌شود، ولی تکنولوژی بکار رفته در آخرین قطعهٔ اتصال (از مرکز تلفن یا ایستگاه توزیع تا مصرف‌کننده) در آنها متفاوت است.

۱-۲ مبانی نظری مخابرات داده

برای انتقال اطلاعات روی سیم می‌توان از متغیرهای الکتریکی مانند ولتاژ یا جریان استفاده کرد. با نمایش تغییرات این ولتاژ یا جریان بصورت تابعی از زمان، می‌توان رفتار سیگنال را مدل‌سازی و تحلیل ریاضی کرد. موضوع این بخش تحلیل ریاضی سیگنالهای الکتریکی است.

۱-۱-۲ آنالیز فوریه

در اوایل قرن نوزدهم میلادی، ریاضیدان فرانسوی ژان باپتیست فوریه ثابت کرد که هر تابع متناوب، g(t)، با دورهٔ تناوب T را می‌توان به صورت مجموع بینهایت جملهٔ سینوسی و کسینوسی نوشت:

$$g(t)=\frac{۱}{۲}c+\sum_{n=۱}^{\infty}a_n\sin(۲\pi nft)+\sum_{n=۱}^{\infty}b_n\cos(۲\pi nft) \qquad (۲-۱)$$

که در آن $f = 1/T$ فرکانس اصلی، a_n و b_n ضرایب **هارمونی** (جملهٔ n ام تابع سینوس و کسینوس، و c یک عدد ثابت است. به تجزیهٔ یک تابع به مجموع بینهایت جملهٔ سینوسی و کسینوسی **سری فوریه** (Fourier series) گفته می‌شود. اگر ضرایب جملات سری فوریهٔ یک تابع و دورهٔ تناوب آن (T) معلوم باشد، می‌توان این تابع را با

استفاده از معادلهٔ (۲-۱) ایجاد کرد.

اگر یک سیگنال داده دارای دورهٔ محدودی باشد (که همیشه هم چنین است)، می‌توان آنرا تکرار بینهایت یک الگوی مشخص فرض کرد (بعبارت دیگر، این موج در فاصلهٔ زمانی T تا $2T$ با فاصلهٔ 0 تا T کاملاً یکسان است). برای محاسبهٔ ضریب a_n کافیست دو طرف معادلهٔ (2-1) را در $\sin(2\pi kft)$ ضرب کرده، و سپس در فاصلهٔ 0 تا T از آن انتگرال بگیریم. از آنجائیکه

$$\int_0^T \sin(2\pi kft)\sin(2\pi nft)\,dt = \begin{cases} 0 & for \quad k\neq n \\ T/2 & for\, k=n \end{cases}$$

فقط جمله‌ای که ضریب a_n دارد، باقی می‌ماند، و جملات b_n بکلی از بین خواهند رفت. به همین ترتیب، اگر دو طرف معادلهٔ (2-1) را در $\cos(2\pi kft)$ ضرب کرده، و سپس در فاصلهٔ 0 تا T از آن انتگرال بگیریم، ضریب b_n بدست خواهد آمد. برای بدست آوردن ثابت c هم کافیست از تابع g(t) در همین فاصلهٔ زمانی انتگرال بگیریم. خلاصهٔ عملیات فوق را در فرمولهای ذیل مشاهده می‌کنید:

$$a_n = \frac{2}{T}\int_0^T g(t)sin(2\pi nft)dt \qquad b_n = \frac{2}{T}\int_0^T g(t)cos(2\pi nft)dt \qquad c = \frac{2}{T}\int_0^T g(t)dt$$

۲-۱-۲　محدودیت پهنای باند

اما برای اینکه ببینید اینها چه ربطی به مخابرات داده دارد، اجازه دهید یک مثال بزنیم: مخابرهٔ کاراکتر اَسکی حرف "b" که بصورت ۸ بیتی کُد شده است. حرف "b" در استاندارد اَسکی ۸ بیتی بصورت 01100010 کُد می‌شود، و اینها بیتهایی هستند که باید مخابره شوند. در نمودار سمت چپ شکل ۲-۱ (الف) ولتاژهای خروجی کامپیوتر برای حرف "b" را می‌بینید. با استفاده از آنالیز فوریه، ضرایب ثابت این سیگنال بصورت زیر بدست می‌آیند:

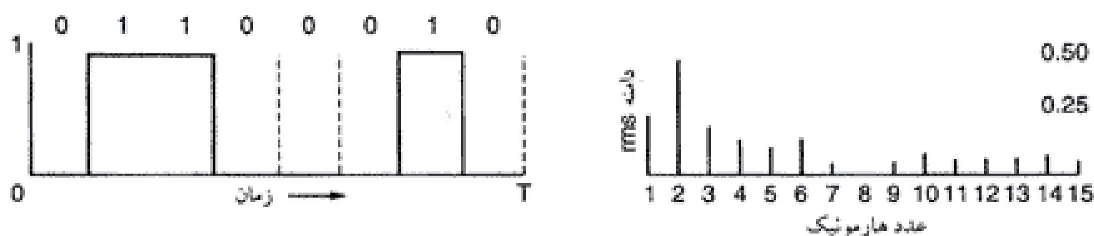
$$a_n = \frac{1}{\pi n}[\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$b_n = \frac{1}{\pi n}[\sin(3\pi n/4) - \sin(\pi n/4) + \sin(7\pi n/4) - \sin(6\pi n/4)]$$

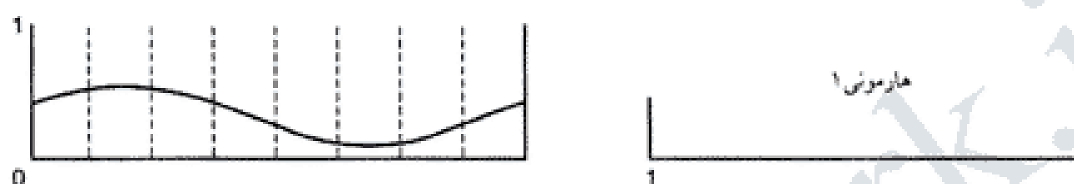
$$c = 3/4$$

در نمودار سمت راست شکل ۲-۱ (الف) مقدار rms ضرایب فوریه، یعنی $\sqrt{a_n^2 + b_n^2}$، چند هارمونی اول این سیگنال را مشاهده می‌کنید. علت اهمیت این ضرایب آنست که انرژی انتشار سیگنال با مجذور آنها (در یک فرکانس خاص) متناسب است.

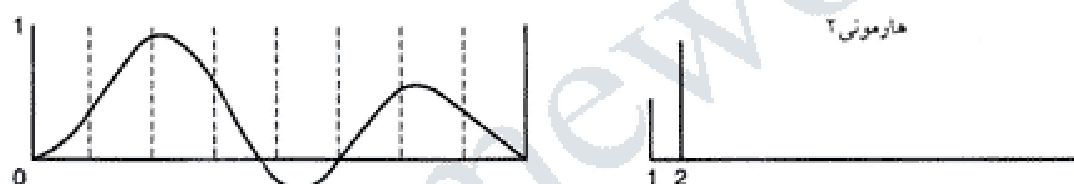
از طرف دیگر می‌دانیم که هیچ سیستم پخشی بدون اتلاف انرژی نیست، ولی اگر تمام ضرایب فوریه به یکسان تضعیف شوند، سیگنال حاصله فقط ضعیفتر می‌شود، و بهیچوجه دچار اعوجاج (تغییر شکل) نخواهد شد. متأسفانه، در سیستمهای پخش مختلف ضرایب فوریه به میزانهای متفاوت تضعیف می‌شوند، و همین باعث بروز اعوجاج در شکل موج خواهد شد. معمولاً، تضعیف موج فقط از فرکانس خاصی به بالا (که به **فرکانس قطع** - f_c - معروف است و با **هرتز** - Hz - یا **سیکل بر ثانیه** - cycles/sec - سنجیده می‌شود) روی می‌دهد، و زیر این فرکانس تضعیف موج وجود ندارد. طیف فرکانسی که موج می‌تواند بدون تضعیف منتشر شود، به **پهنای باند** (bandwidth) معروف است. از آنجائیکه در عمل فرکانس قطع عددی دقیق و قطعی نیست، پهنای باند بعنوان فرکانسی که در آن فقط نیمی از انرژی موج عبور می‌کند، تعریف می‌شود.



(الف)



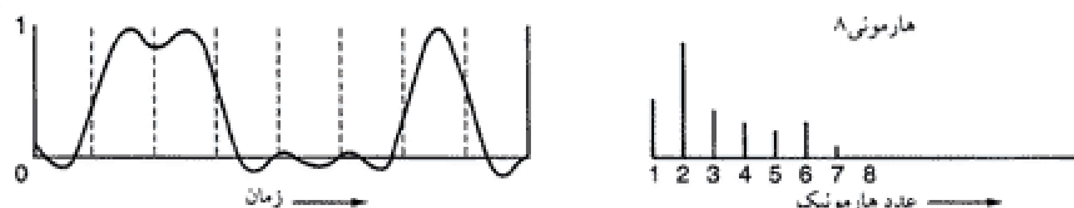
(ب)



(ج)



(د)



(هـ)

شکل ۲-۱. (الف) یک سیگنال باینری و ضرایب فوریهٔ rms آن. (ب)-(هـ) تقریب‌های متوالی سیگنال اولیه.

پهنای باند یکی از خواص فیزیکی رسانهٔ انتقال است، و معمولاً به نوع، شکل، ضخامت و طول آن بستگی دارد. در برخی موارد با قرار دادن یک فیلتر پهنای باندی را که در اختیار مشتری است، عملاً محدود می‌کنند. برای مثال، پهنای باند یک خط تلفن در فواصل کوتاه می‌تواند به 1 MHz برسد، ولی شرکتهای تلفن با قرار دادن

فیلترهای ویژه آنرا به حدود 3100 Hz محدود می‌کنند. این پهنای باند برای انتقال واضح و مفهوم صدا کافیست، و از اتلاف منابع نیز جلوگیری می‌کند.

حال اجازه دهید ببینیم اگر پهنای باند خط انتقال آنقدر کم باشد که فقط فرکانسهای پائین بتوانند منتقل شوند (بعبارت دیگر فقط هارمونی‌های اول معادلهٔ (2-1) اجازه عبور داشته باشند)، سیگنال شکل ۲-۱ (الف) به چه صورتی در می‌آید. در شکل ۲-۱ (ب) شکل موج را در حالتی که فقط هارمونی اول (هارمونی اصلی، f) اجازه عبور دارد، می‌بینید. در شکلهای ۲-۱ (ج)-(هـ) نیز شکل موج برای پهنای باندهای بالاتر نشان داده شده‌اند.

اگر نرخ انتقال اطلاعات b bits/sec باشد، زمان لازم برای ارسال (مثلاً) ۸ بیت معادل $8/b$ sec است، بنابراین فرکانس هارمونی اصلی $b/8$ Hz خواهد بود. همانطور که گفتیم، فرکانس یک خط تلفن معمولی (که به خط ردهٔ **صوتی** – voice-grade – معروف است) بصورت مصنوعی کمی بالای 3000 Hz نگه داشته می‌شود. این محدودیت باعث می‌شود تا بالاترین هارمونی که می‌تواند از این خط عبور کند، تقریباً $3000/(b/8)$ یا $24000/b$ باشد.

در شکل ۲-۲ اعداد محاسبه شده برای برخی از مهمترین نرخهای انتقال را می‌بینید. همانطور که از این شکل برمی‌آید، اگر بخواهیم داده‌ها را با سرعت 9600 bps روی یک خط ردهٔ صوتی بفرستیم، موج ما از شکل ۲-۱ (الف) به شکل ۲-۱ (هـ) تبدیل خواهد شد – که باعث می‌شود تشخیص بیتهای اولیه قدری مشکل شود. از همینجا پیداست که برای سرعتهای بالای 38.4 kbps هیچ شانسی برای ارسال داده‌های باینری وجود ندارد (حتی اگر خط ما کاملاً بدون نویز باشد). بعبارت دیگر، محدود کردن پهنای باند باعث محدود شدن نرخ انتقال اطلاعات (حتی در بهترین کانالها) خواهد شد. با این حال، راههایی وجود دارد (مثلاً، استفاده از چند سطح ولتاژ بجای دو سطح ولتاژ) که می‌توان به نرخهای انتقال بالاتری دست یافت، و در ادامهٔ همین فصل دربارهٔ آنها صحبت خواهیم کرد.

۲-۱-۳ حداکثر نرخ داده در یک کانال

در سال ۱۹۲۴ یکی از مهندسان AT&T بنام هنری نایکوئیست دریافت که حتی بهترین کانال انتقال هم ظرفیت محدودی دارد، و توانست معادلهٔ حداکثر نرخ انتقال دادهٔ یک کانال بدون نویز را برای یک پهنای باند مشخص بدست آورد. در سال ۱۹۴۸ کلود شانون کار نایکوئیست را تکمیل کرد، و معادلهٔ وی را برای حالتی که کانال در معرض نویز تصادفی (نویز ترمودینامیک یا حرارتی) است، توسعه دهد (Shannon, 1948) . در این قسمت کارهای این دو دانشمند را بطور مختصر بررسی خواهیم کرد.

Bps	T (msec)	اولین‌هارمونی (Hz)	تعداد هارمونی فرستاده شده
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

شکل ۲-۲. رابطهٔ بین نرخ انتقال داده و هارمونی‌ها.

نایکوئیست ثابت کرد که اگر سیگنالی از یک **فیلتر پائین‌گذر** (low-pass filter) با پهنای باند H عبور داده شود، سیگنال خروجی (فیلتر شده) را می‌توان با داشتن فقط $2H$ نمونه در ثانیه بازسازی کرد. نمونه‌برداری از خط با نرخ بالاتر از $2H$ دقت بیشتری بهمراه نخواهد داشت، چون فرکانسهای بالاتری که بدین ترتیب می‌توان بازیابی کرد، قبلاً توسط فیلتر حذف شده‌اند. اگر این سیگنال دارای V سطح مجزا باشد، طبق قضیهٔ نایکوئیست:

$$\text{maximum data rate} = 2H \log_2 V \text{ bits/sec}$$

برای مثال، یک کانال 3-kHz بدون نویز نمی‌تواند سیگنالهای باینری (دو سطح ولتاژ) بالاتر از 6000 bps را انتقال دهد.

تا اینجا کانالها را بدون نویز فرض کردیم، که در واقع چنین نیست. اگر در کانال انتقال نویز تصادفی (حرارتی) وجود داشته باشد، اوضاع بسرعت رو به وخامت خواهد گذاشت (و می‌دانیم که هیچ گریزی از نویز حرارتی، که حاصل حرکت مولکولهاست، نیست). مقدار نویز حرارتی با نسبت توان سیگنال به توان نویز سنجیده می‌شود، و به **نسبت سیگنال به نویز** (signal-to-noise ratio) معروف است. اگر توان سیگنال را با S و توان نویز را با N نشان دهیم، نسبت سیگنال به نویز S/N خواهد بود. معمولاً آنچه که به عنوان نسبت سیگنال به نویز داده می‌شود، نه خود S/N بلکه $10 \log_{10} S/N$ است، که به **دسی‌بل** (decibel) معروف است و با dB نشان داده می‌شود. اگر نسبت سیگنال به نویز 10 باشد، معادل 10 دسی‌بل خواهد بود، نسبت 100 معادل 20 دسی‌بل است، نسبت 1000 معادل 30 دسی‌بل، و الی آخر. سازندگان تقویت‌کننده‌های استریو اغلب پهنای باندی را که دستگاه آنها بصورت خطی عمل می‌کند، بصورت فرکانس 3 دسی‌بل در هر انتهای طیف مشخص می‌کنند (فرکانس 3 دسی‌بل فرکانسی است که ضریب تقویت دستگاه نصف می‌شود، چون $\log_{10}3 = 0.5$).

شانون معادلهٔ نایکوئیست را برای کانالهای نویزدار بصورت زیر تصحیح کرد:

$$\text{maximum number of bits/sec} = H \log_2 (1 + S/N)$$

که در آن H پهنای باند کانال (بر حسب هرتز)، و S/N نسبت سیگنال به نویز است. برای مثال، در کانالی با پهنای باند 3000 Hz که دارای نویز حرارتی 30 dB است (نویز معمول در بخش آنالوگ سیستمهای تلفن)، هرگز نمی‌توان بیش از 30,000 bps داده ارسال کرد (بدون توجه به اینکه تعداد سطوح ولتاژ این سیگنال یا تعداد نمونه‌برداری‌ها چقدر باشد).

معادلهٔ شانون از **تئوری اطلاعات** (information theory) استنتاج شده، و برای هر کانالی که تحت تأثیر نویز حرارتی باشد صادق است. کانالی که این قاعده را نقض کند، باید معادل یک ماشین حرکت دائم فرض کرد (ماشینی که طبق اصول ترمودینامیک نمی‌تواند وجود خارجی داشته باشد). البته باید توجه داشته باشید که این عدد فقط حد بالای ظرفیت کانال را مشخص می‌کند، و سیستمهای واقعی بندرت به این حد دست پیدا می‌کنند.

۲-۲ رسانهٔ انتقال هدایت‌پذیر

وظیفهٔ لایهٔ فیزیکی انتقال بیت‌های خام از یک ماشین به ماشین دیگر است. برای اینکار از رسانه‌های فیزیکی مختلفی می‌توان استفاده کرد، که هر کدام پهنای باند، تأخیر انتشار، هزینه، و سهولت نصب و نگهداری خاص خود را دارند. این رسانه‌ها را می‌توان به دو دستهٔ کلی هدایت‌پذیر (مانند سیم مسی یا فیبر نوری) و هدایت‌ناپذیر (مانند امواج رادیویی و لیزری) تقسیم‌بندی کرد. در قسمتهای آینده این رسانه‌ها را مورد بررسی قرار خواهیم داد.

۲-۲-۱ رسانهٔ مغناطیسی

یکی از متداولترین راههای انتقال اطلاعات از کامپیوتری به کامپیوتر دیگر، نوشتن آنها روی نوار مغناطیسی، دیسک، CD یا DVD و سپس خواندن آنها در کامپیوتر مقصد است. اگرچه این روش بخوبی استفاده از مخابرات

ماهواره‌ای نیست، ولی هزینهٔ آن بسیار کمتر است، بویژه در مواردی که پهنای باند زیاد و قیمت کم جزء عوامل کلیدی باشند.

اجازه دهید با یک مثال ساده مطلب را روشن‌تر کنیم. امروزه نوارهای مغناطیسی با ظرفیت 200 GB بوفور در بازار یافت می‌شوند، و یک جعبهٔ مقوایی به ابعاد 60 cm × 60 × 60 می‌تواند 1000 تا از این نوارها را در خود جای دهد، که بدین ترتیب ظرفیت آن به 200 TB (ترابایت، معادل هزار گیگابایت) یا 1600 Tbit (که معادل 1.6 Pbit – پتابیت – است) می‌رسد. این بسته را می‌توان با استفاده از پست سریع‌السیر در کمتر از ۲۴ ساعت به هر نقطه‌ای در ایالات متحدهٔ آمریکا تحویل داد. بدین ترتیب، پهنای باند مؤثر این سیستم انتقال 1600 terabits/86,400 sec یا 19 Gbps است – و اگر فاصلهٔ این دو نقطه فقط یک ساعت باشد، پهنای باند تا 400 Gbps نیز افزایش خواهد یافت. هیچ شبکهٔ کامپیوتری حتی نمی‌تواند به این سرعت نزدیک شود.

برای شبکه‌های بانکی که در هر روز باید چندین گیگابایت اطلاعات را از نقطه‌ای به نقطهٔ دیگر منتقل کنند (تا در صورت بروز فجایع طبیعی نیز بتوانند به کار خود ادامه دهند) هیچ چیز نمی‌تواند جای نوار مغناطیسی را بگیرد. البته شبکه‌ها هر روز سریعتر می‌شوند، ولی ظرفیت نوارهای مغناطیسی نیز رو به افزایش است.

در مورد هزینه نیز وضعیت مشابهی وجود دارد. قیمت هر نوار مغناطیسی 200 GB چیزی حدود 40 دلار است (البته در خریدهای عمده)، و از هر نوار می‌توان حداقل ۱۰ بار استفاده کرد. بنابراین، هزینهٔ هر بار استفاده از جعبه نوار ۱۰۰۰ تایی 4000 دلار خواهد شد. اگر 1000 دلار نیز برای حمل و نقل به آن اضافه کنیم (که البته معمولاً بسیار کمتر است)، هزینهٔ کل به 5000 دلار برای انتقال 200 TB اطلاعات می‌رسد – و این یعنی ۳ سنت (معادل 0.01 دلار) برای هر گیگابایت، که هیچ شبکه‌ای نمی‌تواند با آن رقابت کند. نتیجهٔ اخلاقی داستان:

> هرگز پهنای باند کامیونی پُر از نوارهای مغناطیسی را که بسرعت در بزرگراه در حال حرکت است، دست کم نگیرید.

۲-۲-۲ زوج تابیده

با اینکه پهنای باند نوار مغناطیسی بسیار عالیست، تأخیر انتشار آن (زمانی که طول می‌کشد تا اولین بیت اطلاعات به مقصد برسد) ناامیدکننده است – در شبکه‌ها معمولاً با میلی‌ثانیه سروکار داریم نه روز و ساعت! در بسیاری از کاربردها برقرار بودن دائمی ارتباط یک امر حیاتی است. یکی از قدیمی‌ترین (و همچنان متداولترین) رسانه‌های انتقال **زوج تابیده** (twisted pair) است. زوج تابیده عبارتست از یک زوج سیم مسی عایق‌دار (بضخامت 1 mm)، که به صورت مارپیچ به دور یکدیگر تابیده‌اند (مانند زنجیرهٔ مولکول DNA). علت تابیدن سیمها آنست که دو سیم مسی معمولی مانند یک آنتن عمل کرده، و انرژی تلف می‌کنند. در حالت تابیده امواج سیمها یکدیگر را خنثی کرده، و تشعشع به حداقل می‌رسد.

بیشترین کاربرد زوج تابیده در شبکه‌های تلفن است: تقریباً تمام تلفنها با استفاده از زوج تابیده به مرکز تلفن وصل می‌شوند. از زوجهای تابیده می‌توان بطول چندین کیلومتر بدون نیاز به تقویت‌کننده استفاده کرد، ولی برای مسافتهای طولانی‌تر به **تکرارکننده** (repeater) نیاز هست. تعداد زیادی زوج تابیده که در یک غلاف محافظ جمع شده باشند، تشکیل یک کابل زوج تابیده را می‌دهند. اگر سیمهای این کابل بصورت دو به دو به یکدیگر تابیده باشند، تداخل شدیدی بین آنها بوجود خواهد آمد.

از زوجهای تابیده برای انتقال سیگنالهای آنالوگ و دیجیتال می‌توان استفاده کرد. پهنای باند این خطوط به ضخامت سیمها و مسافت بستگی دارد، و در فواصل کوتاه (دو تا سه کیلومتر) می‌توان به پهنای باند چندین مگابیت بر ثانیه دست یافت. بدلیل کارایی کافی و هزینهٔ پائین، از زوج تابیده بنحو گسترده‌ای استفاده شده است، و بنظر می‌رسد تا سالها نیز این وضعیت ادامه یابد.



شکل ۲-۳. (الف) زوج تابیدهٔ Cat 3 . (ب) زوج تابیدهٔ Cat 5 .

انواع مختلفی از کابلهای زوج تابیده وجود دارد، که دو تا از آنها در شبکه‌های کامپیوتری از اهمیت بیشتری برخوردار هستند. در کابلهای Category 3 (که به Cat 3 نیز معروف است) سیمها با شدت کمتری به هم تابیده‌اند. کابل Cat 3 از چهار زوج تابیده، که در یک غلاف پلاستیکی قرار می‌گیرند، تشکیل می‌شود. تا قبل از سال ۱۹۸۸ تقریباً در همه جا از این نوع کابل استفاده می‌شد. کابل Cat 3 می‌توانست چهار خط تلفن معمولی (یا دو خط تلفن مرکب) را از ایستگاه تلفن به نقطهٔ موردنظر برساند.

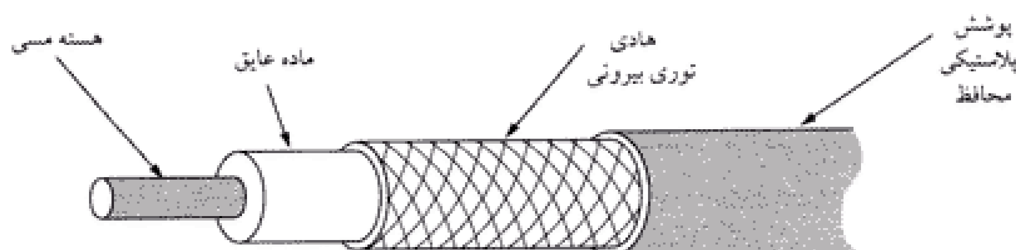
در سال ۱۹۸۸ کابل پیشرفته‌تر Category 5 (یا همان Cat 5) وارد بازار شد. این کابل شبیه Cat 3 است، با این تفاوت که تعداد دورهای آن در واحد طول بیشتر بوده، و بهمین دلیل تداخل سیگنال در آن کاهش یافته و برای شبکه‌های پُرسرعت مناسبتر است. کابلهای Cat 6 و Cat 7 نیز در حال آمدن به بازار هستند، که سرعت آنها بترتیب به 250 MHz و 600 MHz می‌رسد (برای مقایسه، سرعت کابلهای Cat 3 و Cat 5 بترتیب 16 MHz و 100 MHz است).

تمام این کابلها در بازار بنام UTP (**زوج تابیدهٔ بدون زره** - Unshielded Twisted Pair) شناخته می‌شوند، تا از کابلهای STP (**زوج تابیدهٔ زره‌دار** - Shielded Twisted Pair) که IBM در اوایل دههٔ ۱۹۸۰ معرفی کرد (و بدلیل گرانی و سختی کار با آن، استقبال چندانی از این نوع کابل نشد)، متمایز باشند. در شکل ۲-۳ دو نوع زوج تابیدهٔ Cat 3 و Cat 5 (و تفاوت آنها) را ملاحظه می‌کنید.

۲-۲-۳ کابل کواکسیال

یکی دیگر از رسانه‌های رایج کابل **کواکسیال** (هم‌محور - coaxial) است (که اغلب به آن **کابل کواکس** می‌گویند). این کابل بدلیل داشتن زره (غلاف فلزی) کارایی بهتری نسبت به زوج تابیده (هم از نظر سرعت، هم از نظر مسافت) دارد. کابل کواکسیال دو نوع دارد. اولی کابل 50-ohm است، که از ابتدا برای مخابرات دیجیتال در نظر گرفته شده؛ و دومی کابل 75-ohm ، که ابتدا برای مخابرات آنالوگ و تلویزیون کابلی بکار می‌رفت، ولی امروزه با گسترش اینترنت کابلی از اهمیت روزافزونی برخوردار شده است. این تمایز بیش از آن که فنی باشد، جنبهٔ تاریخی دارد: امپدانس آنتنهای دوقطبی قدیمی 300-ohm بود، و ترانسفورماتورهای تطبیق امپدانس 4:1 بوفور در بازار یافت می‌شد.

کابل کواکسیال تشکیل می‌شود از یک سیم مسی سخت بعنوان **هسته** (core) ، یک لایه عایق استوانه‌ای به دور این هسته، یک لایه توری فلزی که به دور عایق بافته شده، و لایهٔ پلاستیکی محافظ خارجی (شکل ۲-۴).



شکل ۲-۴. ساختمان کابل کواکسیال.

ساختمان و نحوهٔ عایق‌بندی کابل کواکسیال باعث شده تا این نوع کابل از نظر سرعت و مصونیت در مقابل نویز کارایی بسیار خوبی داشته باشد. پهنای باند کابلهای کواکسیال به کیفیت مواد آن، طول کابل و نسبت سیگنال به نویز امواج ارسالی بستگی دارد، و در کابلهای جدید به 1 GHz نیز می‌رسد. از کابل کواکسیال بیشتر در سیستمهای تلفن راه دور استفاده می‌شد، که امروزه بتدریج جای خود را به فیبرهای نوری می‌دهد. با این حال، در شبکه‌های شهری و تلویزیون کابلی هنوز هیچ رقیبی برای کابل کواکس وجود ندارد.

۲-۲-٤ فیبر نوری

بسیاری از افرادی که در صنعت کامپیوتر کار می‌کنند، از شتاب خیره‌کنندهٔ رشد تکنولوژی آن به خود می‌بالند. اولین کامپیوتر شخصی (PC) که IBM در سال ۱۹۸۱ به بازار عرضه کرد، با سرعت ساعت 4.77 MHz کار می‌کرد؛ بعد از گذشت بیست سال، امروزه PC ها می‌توانند با سرعتی متجاوز از 2 GHz کار کنند، و این یعنی ۲۰ برابر شدن سرعت در هر ۱۰ سال - چندان هم بد نیست!

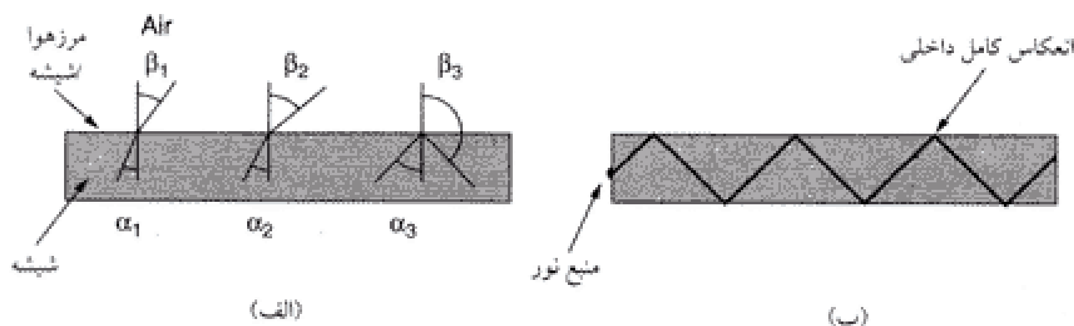
در همین دوره، سرعت مخابرات راه دور از 56-kbps (آرپانت) به 1 Gbps (مخابرات نوری جدید) رسیده است - این یعنی رشدی معادل ۱۲۵ برابر در هر دهه (در حالیکه میزان خطا هم از یک بیت در هر 10^5 بیت به تقریباً صفر رسیده).

اما پیشرفت هم مرزهای فیزیکی خاص خود را دارد؛ CPU ها مدتیست با موانعی از قبیل محدودیت سرعت نور و مشکل اتلاف گرما روبرو هستند. در حالیکه این مسائل در مخابرات وجود ندارد، و تکنولوژی فیبر نوری می‌تواند براحتی به پهنای باندی فراتر از 50,000 Gbps (یا 50 Tbps) دست یابد (و از هم اکنون بسیاری افراد با جدیت بدنبال تکنولوژیهای بهتر نیز هستند). سرعت فعلی مخابرات فیبر نوری به چیزی در حدود 10 Gbps محدود می‌شود، ولی علت اصلی آن محدودیت سرعت تبدیل سیگنالهای الکتریکی به پالسهای نوری است، نه محدودیت ذاتی فیبرهای نوری (البته در آزمایشگاهها به سرعت 100 Gbps نیز دست یافته‌اند).

در مسابقهٔ بین کامپیوتر و مخابرات، مسلماً مخابرات برنده است. البته هنوز عدهٔ زیادی از دانشمندان و مهندسان کامپیوتر (که به قانون نایکوئیست و شانون فکر می‌کنند، و نمی‌دانند که این قانونها مربوط به سیمهای مسی است) نمی‌توانند تصور پهنای باند نامحدود را به ذهن خود راه دهند. قانون آینده باید این باشد که سرعت کامپیوترها هرگز به گرد شبکه‌ها نیز نخواهد رسید، و باید از هر نوع پردازشی روی اطلاعات شبکه خودداری کرد (حتی اگر به معنای تلف شدن مقداری از پهنای باند باشد). اما اجازه دهید ببینیم فیبر نوری چگونه کار می‌کند.

یک سیستم انتقال نوری سه مولفهٔ کلیدی دارد: منبع نور، رسانهٔ انتقال، آشکارساز. طبق قرارداد، یک پالس نوری معادل بیت 1 و فقدان نور معادل بیت 0 است. رسانهٔ انتقال یک رشته (فیبر) فوق‌العاده نازک شیشه است. **آشکارساز** (detector) نیز دستگاهیست که با برخورد نور به آن یک پالس الکتریکی تولید می‌کند. با قرار دادن یک منبع نور در یک سر فیبر نوری و یک آشکارساز در سمت دیگر، یک سیستم انتقال نوری یکطرفه خواهیم داشت که سیگنال الکتریکی را گرفته، آنرا به پالسهای نوری تبدیل کرده، و در طرف دیگر پالس دریافتی را به سیگنالهای الکتریکی تبدیل می‌کند.

اما می‌دانیم که نور از شیشه خارج می‌شود، و چنین سیستمی بکلی بی‌فایده خواهد بود. اینجاست که یکی از قوانین جالب فیزیک نور به کمک ما می‌آید. این قانون (که به **قانون شکست نور** معروف است) می‌گوید که وقتی پرتو نور از یک محیط (مثلاً، شیشه) وارد محیط دیگر (مثلاً، هوا) می‌شود، در مرز این دو محیط دچار خمیدگی یا **شکست** (refraction) می‌شود. به شکل ۲-۵ (الف) نگاه کنید؛ در این شکل یک پرتو نور می‌بینید که با زاویهٔ α_1 به مرز شیشه و هوا برخورد کرده، و با زاویهٔ β_1 از شیشه خارج و وارد هوا می‌شود. مقدار خمیدگی یا شکست پرتو نور به خواص فیزیکی دو محیط (بویژه **ضریب شکست** آنها) بستگی دارد. اگر زاویهٔ برخورد نور از یک مقدار



شکل ۲-۵. (الف) سه مثال از پرتوهای نوری که به مرز شیشه-هوا برخورد کرده، و شکسته می‌شوند. (ب) پرتو نور بدلیل شکست کلی در داخل شیشه گرفتار شده است.

بحرانی بیشتر باشد، پرتو نور دچار **شکست کلی** شده و دوباره به داخل شیشه برمی‌گردد، و هرگز وارد هوا نخواهد شد. نوری که با این زاویه (یا بیشتر از آن) به داخل شیشه تابانده شود، برای همیشه در آن محبوس می‌شود، و می‌تواند مسافتهای طولانی را بدون اتلاف انرژی در فیبر نوری بپیماید - شکل ۲-۵(ب) را ببینید.

در شکل ۲-۵(ب) فقط یک پرتو نور نشان داده شده است، ولی از آنجائیکه هر پرتو نوری که با زاویهٔ بالاتر از زاویهٔ بحرانی به مرز شیشه و هوا برخورد کند به داخل شیشه برمی‌گردد، در هر لحظه پرتوهای متعددی با زاویه‌های مختلف در داخل فیبر نوری به بالا و پائین حرکت می‌کنند، و اصطلاحاً گفته می‌شود که هر یک از این پرتوها دارای **حالت** (mode) خاص خود است. به فیبری که چنین ویژگی داشته باشد، **فیبر چندحالته** (multimode fiber) گفته می‌شود.

ولی اگر قطر فیبر فقط چند برابر طول موج نور باشد، پرتو نور فقط می‌تواند در جهت مستقیم (بدون جهش به بالا و پائین) حرکت کند. به چنین فیبری که بصورت یک **موج‌بَر** (wave guide) عمل می‌کند، **فیبر تک‌حالته** (single-mode fiber) گفته می‌شود. فیبرهای تک‌حالته گرانتر از فیبرهای چندحالته هستند، ولی در مسافتهای طولانی اغلب از این نوع فیبرها استفاده می‌شود. فیبرهای تک‌حالتهٔ امروزی می‌توانند با ظرفیتهایی تا 50 Gbps و بطول 100 کیلومتر بدون نیاز به تقویت‌کننده کار کنند. (در آزمایشگاهها برای مسافتهای کوتاهتر به پهنای باند بالاتری نیز دست یافته‌اند.)

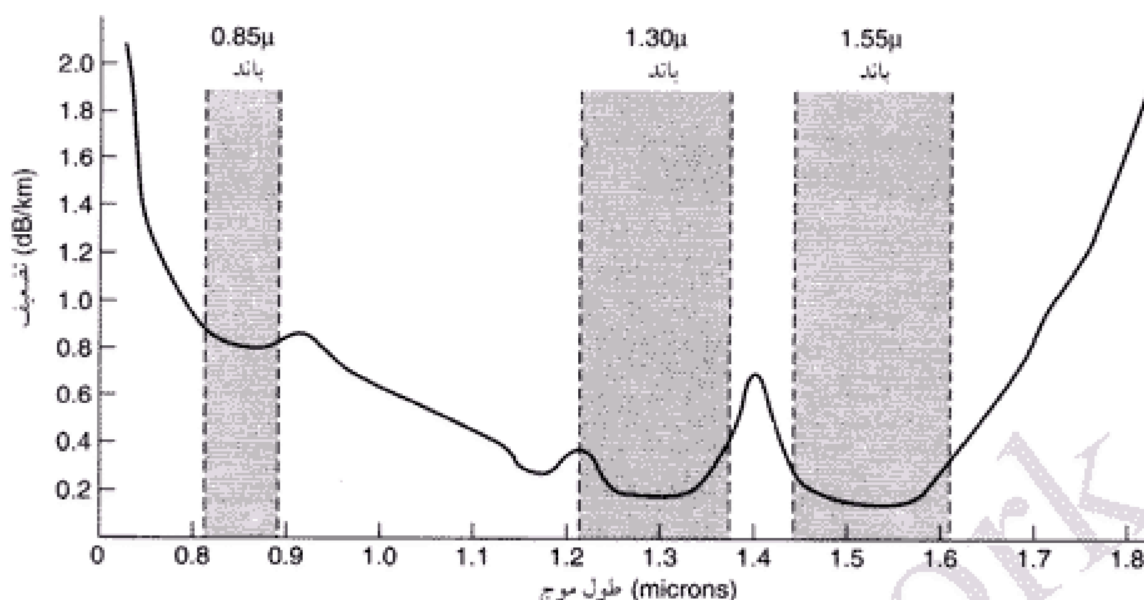
عبور نور در درون فیبر نوری

فیبرهای نوری از شیشه ساخته می‌شوند و خود شیشه هم از شن، ماده‌ای که بوفور در طبیعت یافت می‌شود. شیشه برای اولین بار در مصر باستان ساخته شد، ولی این شیشه‌ها بقدری کدر بودند که در ضخامتهای بیشتر از 1 mm نور را از خود عبور نمی‌دادند. شیشه‌ای که برای استفاده در پنجره‌ها مناسب باشد، در دورهٔ رنسانس ساخته شد. شیشه‌ای که در ساخت فیبرهای نوری بکار می‌رود آنقدر شفاف است که اگر اقیانوس را با آن پُر کنیم، کف آن بوضوح دیده خواهد شد.

تضعیف نور (attenuation) در شیشه به طول موج آن و برخی از خواص فیزیکی شیشه بستگی دارد. در شکل ۲-۶ میزان تضعیف نور در فیبرهای نوری بر حسب دسی‌بل بر کیلومتر (خطی) نشان داده شده است. معادلهٔ تضعیف نور چنین است:

$$\text{attenuation (dB)} = 10 \log_{10} \frac{\text{transmitted power}}{\text{received power}}$$

برای مثال، اگر قدرت دریافتی در خروجی نصف قدرت ورودی باشد، میزان تضعیف سیگنال $10 \log_{10} 2 = 3$ dB است. در شکل ۲-۶ نمودار تضعیف نور را در ناحیهٔ مادون‌قرمز طیف، که در عمل نیز از آن



شکل ۲-۶. تضعیف نور عبوری از فیبر نوری در ناحیهٔ مادون‌قرمز.

استفاده می‌شود، می‌بینید. طول موج نور مرئی (0.4-0.7 میکرون، یا 400-700 نانومتر) قدری کوتاهتر از نور مادون‌قرمز است ($1\ nm = 10^{-9}\ m$ و $1\ \mu m = 10^{-6}\ m$).

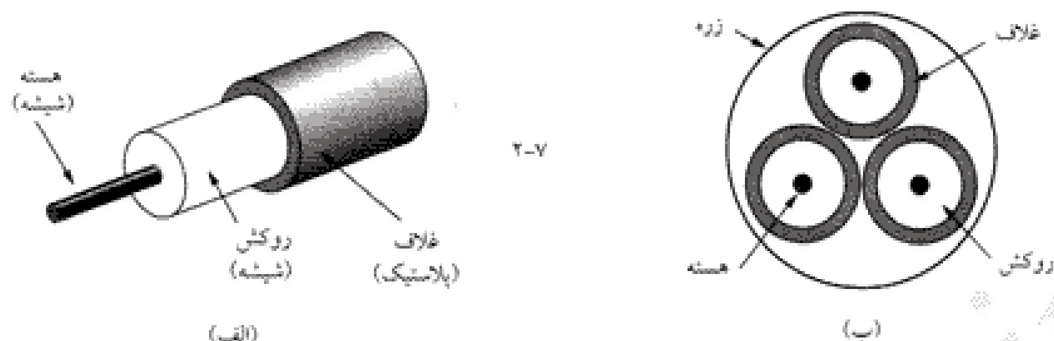
برای مخابرات نوری سه باند از طول موجهای نور مادون‌قرمز مورد استفاده قرار می‌گیرد، که بترتیب حول طول موجهای 0.85 ، 1.30 و 1.55 میکرون متمرکز شده‌اند. دو باند آخر دارای مشخصات تضعیف خوبی (کمتر از ۵ درصد در هر کیلومتر) هستند. باند 0.85 میکرون میزان تضعیف بالاتری دارد، ولی در این طول موج می‌توان تجهیزات نوری و الکترونیکی را از یک مادهٔ نیمه‌هادی واحد (گالیوم آرسنید) ساخت. پهنای این باندها همگی بین 25,000 GHz تا 30,000 GHz است.

وقتی پالسهای نور از فیبر نوری عبور می‌کنند، پهنای آنها زیاد می‌شود، که این پدیده به **پراکنش کروماتیک** (chromatic dispersion) معروف است، و مقدار آن به طول موج نور بستگی دارد. یکی از راههای جلوگیری از تداخل این پالسهای پهن شده، افزایش فاصلهٔ آنهاست، و این کار نیز فقط با کاهش نرخ ارسال سیگنال ممکن است. خوشبختانه با کشف این موضوع که می‌توان با تولید پالسهایی که تقارن کسینوس هذلولی دارند، اثر پراکنش کروماتیک را بکلی حذف کرد، امکان آن بوجود آمده که پالسهای نوری را بدون تغییر شکل محسوس تا مسافتهای هزاران کیلومتری مخابره کرد. امروزه تلاشهای زیادی در دست انجام است تا این پالسها را، که به آنها **سولیتون** (soliton) گفته می‌شود، از حالت آزمایشگاهی خارج و در عمل بکار بگیرند.

کابل فیبر نوری

کابل فیبر نوری بسیار شبیه کابل کواکسیال است، با این تفاوت که غلاف توری فلزی بیرونی را ندارد (به شکل ۲-۷ (الف) نگاه کنید). رشتهٔ شیشه‌ای که نور از آن عبور می‌کند، در مرکز کابل قرار دارد. در فیبرهای چندحالته قطر این رشته معمولاً $50\ \mu m$ (تقریباً معادل ضخامت موی سر انسان) است، و در فیبرهای تک‌حالته قطر آن به 8 تا 10 میکرون می‌رسد.

هستهٔ فیبر نوری با یک روکش شیشه‌ای (با ضریب شکست کمتر) پوشانده می‌شود، تا تمام نور در رشتهٔ مرکزی باقی بماند. پوشش بیرونی نیز (که اغلب پلاستیکی است) نقش محافظ را بازی می‌کند. معمولاً چند کابل تک رشته را در یک غلاف گرد هم می‌آورند (شکل ۲-۷ (ب) را ببینید).



شکل ۲-۷. (الف) نمای کناری یک فیبر منفرد. (ب) سطح مقطع کابلی با سه فیبر.

کابلهای زمینی معمولاً در عمق یک متری سطح زمین دفن می‌شوند (که در آنجا از آسیب بیلهای مکانیکی یا موشهای جونده در امان نیستند). در قسمتهای کم عمق ساحل، کابلهای زیردریایی را در کانالهای مخصوص پنهان می‌کنند، ولی در آبهای عمیق (که کندن کانال عملی نیست) آنها را آزاد در کف دریا رها می‌کنند (و با این کار آنها را در معرض آسیب از طرف کشتی‌های ماهیگیری و اسکوئیدهای غول‌پیکر قرار می‌دهند).

سه روش برای متصل کردن فیبرهای نوری وجود دارد. در روش اول، به انتهای کابل پایانه‌های مخصوص وصل کرده، و آنها را به سوکت فیبر نوری متصل می‌کنند. این پایانه‌ها ۱۰ تا ۲۰ درصد نور را تلف می‌کنند، ولی کار با آنها بسیار ساده است.

دوم اینکه، می‌توان رشته‌ها را بطور مکانیکی بهم متصل کرد. در این روش دو سر رشته‌ها را که با دقت بریده شده‌اند، در یک غلاف روبروی هم قرار می‌دهند، و آنها را در جای خود محکم می‌کنند. در این روش با عبور دادن نور و تنظیم رشته‌ها می‌توان به حداکثر سیگنال عبوری دست یافت. اتلاف نور در این روش فقط ۱۰ درصد است، و یک فرد تعلیم دیده می‌تواند در عرض ۵ دقیقه چنین اتصالی را بوجود آورد.

در روش سوم، دو سر رشته‌ها ذوب و در هم فرو برده می‌شود، تا یک اتصال یکپارچه بوجود آید. این بهترین نوع اتصال است، چون رشته‌ها در واقع یکی می‌شوند، ولی حتی در این روش هم مقداری افت توان وجود دارد.

در هر سه روش فوق، محل اتصال می‌تواند مقداری از نور را بازتابش کند، که این نور با سیگنال اصلی تداخل خواهد کرد.

تبدیل سیگنالهای الکتریکی به پالسهای نوری معمولاً به دو روش صورت می‌گیرد: لیزرهای نیمه‌هادی و LED (Light Emitting Device). هر کدام از این منابع نوری ویژگیهای خاص خود را دارند (به شکل ۲-۸ نگاه کنید). طول موج یک منبع نور را می‌توان با قرار دادن **تداخل‌سنج** (interferometer) هایی از نوع **فابری-پروت** (Fabry-Perot) یا **ماخ-زندر** (Mach-Zehnder) بین منبع نور و فیبر نوری تنظیم کرد. تداخل‌سنج فابری-پروت یک **حفرهٔ تشدید** (resonant cavity) ساده است، که از دو آینهٔ موازی تشکیل می‌شود. نور بصورت عمود به این آینه‌ها تابانده می‌شود، و فقط طول‌موجهایی که مضرب صحیحی از طول حفره باشند، می‌توانند از آن خارج شوند. در تداخل‌سنج ماخ-زندر نور به دو پرتو جداگانه تقسیم می‌شود، که پس از طی مسافتی کوتاه دوباره با هم ترکیب می‌شوند؛ این دو پرتو فقط در طول موجهای خاصی با یکدیگر همفاز هستند.

در انتهای دیگر فیبر نوری یک **فتودیود** (photodiode) قرار می‌گیرد، که با هر پالس نوری یک سیگنال الکتریکی تولید می‌کند. زمان پاسخ این نوع دیودها معمولاً 1 nsec است ($1\ nsec = 10^{-9}\ sec$)، که باعث می‌شود نرخ داده‌ها به 1 Gbps محدود شود. از آنجائیکه نویز حرارتی در اینجا هم می‌تواند باعث بروز مشکل می‌شود، پالس نوری باید آنقدر انرژی داشته باشد که بتوان آنرا از نویز تشخیص داد (در فیبرهای نوری می‌توان ضریب خطا را با افزایش انرژی پالسهای نوری بمیزان دلخواه پائین آورد).

آیتم	LED	لیزر نیمه هادی
سرعت داده	کم	زیاد
نوع فیبر	چند حالته	چند حالته یا تک حالته
فاصله	کوتاه	بلند
طول عمر	زیاد	کم
حساسیت به دما	کم	قابل توجه
قیمت	کم	زیاد

شکل ۲-۸. مقایسه‌ای بین لیزرهای نیمه‌هادی و LED بعنوان منبع نور.

شبکه‌های فیبر نوری

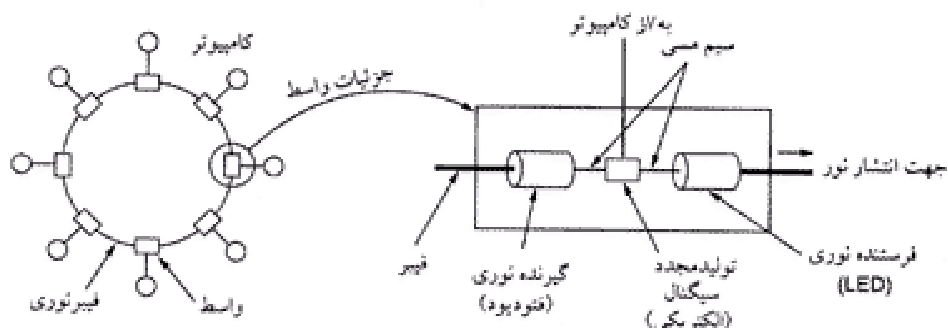
از فیبر نوری، علاوه بر مخابرات راه دور، در شبکه‌های محلی نیز می‌توان استفاده کرد - اگرچه این کار نسبت به اترنت با مشکلات بیشتری همراه است. یک شبکهٔ حلقوی را در نظر بگیرید (شکل ۲-۹) - این شبکه در واقع مجموعه‌ایست از چند اتصال نقطه-به-نقطه. هر کامپیوتر یک **اتصال T** (T junction) به شبکه دارد، که می‌تواند پالسهای نور را از خود عبور دهد، یا آنها را دریافت کند.

دو نوع اتصال وجود دارد: **غیرفعال** (passive) و **فعال** (active) . در اتصال غیرفعال دو توپی به فیبر اصلی متصل می‌شود، که یکی LED یا دیود لیزری دارد (برای ارسال)، و دیگری فتودیود (برای دریافت). خود توپی هیچ عنصر فعالی ندارد و به همین دلیل فوق‌العاده قابل‌اعتماد است، چون خراب شدن LED یا فتودیود باعث قطعی شبکه نخواهد شد و فقط همان یک کامپیوتر از شبکه قطع می‌شود.

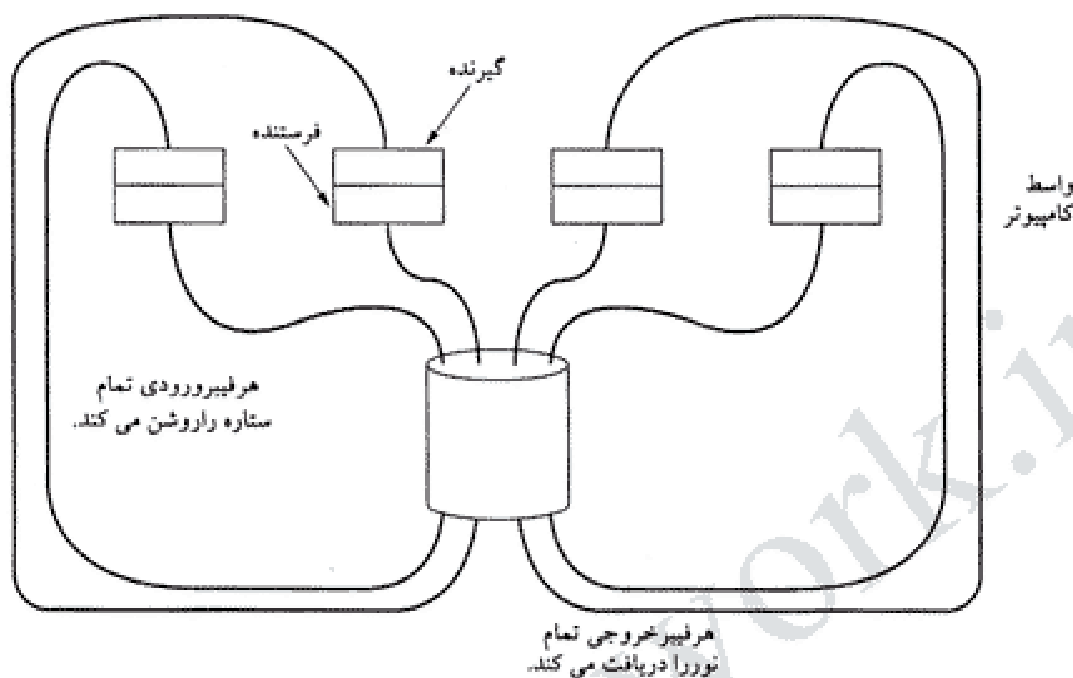
اتصال دیگری که در شکل ۲-۹ می‌بینید، **تکرارکنندهٔ فعال** (active repeater) است. در اینجا، نور ورودی ابتدا به سیگنال الکتریکی تبدیل شده، در صورت نیاز تقویت و دوباره بصورت نور منتشر می‌شود. اتصال به کامپیوتر توسط یک سیم مسی ساده که از تقویت کنندهٔ سیگنال منشعب شده، برقرار می‌شود. امروزه تقویت‌کننده‌های تمام نوری (که مرحلهٔ تبدیل به سیگنال الکتریکی - و بالعکس - در آنها حذف شده) نیز به بازار آمده‌اند؛ پهنای باند این تکرارکننده‌ها بسیار بالاست.

اگر یک تکرارکنندهٔ فعال خراب شود، حلقه شکسته شده و کل شبکه از کار می‌افتد. از طرف دیگر به علت تقویت سیگنال، فاصلهٔ کامپیوترها در این شبکه می‌تواند به چندین کیلومتر برسد، و از نظر تعداد کامپیوترها هم هیچ محدودیتی وجود ندارد. (در نوع غیرفعال، هر اتصال مقداری از توان نوری را هدر می‌دهد، بنابراین تعداد کامپیوترها و طول کل حلقه بشدت محدود است.)

توپولوژی حلقه تنها راه برای ایجاد شبکه‌های محلی با فیبر نوری نیست، و می‌توان به کمک سخت‌افزارهای خاص (انتشاردهنده‌های نوری) ساختاری موسوم به **ستارهٔ غیرفعال** (passive star) بوجود آورد (شکل ۲-۱۰).



شکل ۲-۹. یک شبکهٔ حلقوی فیبر نوری، با تکرارکننده‌های فعال.



شکل ۲-۱۰. شبکه‌ای با ساختار ستارهٔ غیرفعال.

انتشاردهندهٔ نوری یک استوانهٔ شیشه‌ای خاص است، که فیبرهای ورودی (گیرنده) به یک طرف و فیبرهای خروجی (فرستنده) به طرف دیگر آن جوش خورده‌اند. وقتی یک کامپیوتر سیگنالی می‌فرستد، پالس نوری حاصله از طریق این استوانه در تمام فیبرهای متصل به گیرنده‌ها پخش می‌شود. در واقع، ستارهٔ غیرفعال پالسهای ورودی را ترکیب کرده، و نور حاصله را روی تمام فیبرها می‌فرستد. از آنجائیکه انرژی نوری دریافتی در ستارهٔ غیرفعال بین تمام رشته‌ها پخش می‌شود، تعداد کامپیوترهای متصل به این شبکه محدود (و وابسته به حساسیت فتودیودها) است.

مقایسهٔ فیبر نوری و سیم مسی

در اینجا بد نیست مقایسه‌ای بین فیبر نوری و سیم مسی داشته باشیم؛ حتماً آموزنده خواهد بود. فیبر نوری مزایای متعددی نسبت به سیم مسی دارد. از همه مهمتر اینکه پهنای باند آن بسیار بیشتر از سیم مسی است، و همین دلیل کافیست تا در شبکه‌های پُرسرعت انتخاب اول باشد. دیگر اینکه بدلیل اتلاف قدرت ناچیز فقط در هر ۵۰کیلومتر به تکرارکننده نیاز دارد، در حالیکه این مسافت برای سیم مسی ۵ کیلومتر بیشتر نیست، که این باعث صرفه‌جویی زیادی در هزینه‌ها خواهد شد. مزیت دیگر فیبر نوری عدم تأثیرپذیری آن نسبت به نویز و تداخل‌های الکترومغناطیسی است. مقاومت عالی شیشه در مقابل خوردگی‌های شیمیایی نیز یکی دیگر از نقاط قوت فیبر نوری محسوب می‌شود.

جالب است بدانید که علت علاقهٔ شرکتهای تلفن به فیبر نوری اساساً چیز دیگریست: نازکی و سبکی. کانالهای زیرزمینی که شرکتهای تلفن به نقاط مختلف حفر کرده‌اند، اکنون تا حد اشباع پُر شده‌اند، و دیگر جای خالی برای انشعابات جدید ندارند. جایگزین کردن کابلهای مسی با فیبر نوری، علاوه بر خالی کردن این کانالها، درآمد خوبی نیز برای این شرکتها بهمراه دارد: فروش مس خالص به پالایشگاههای مس. علاوه بر آن، فیبرهای نوری بسیار سبکتر از سیمهای مسی هستند: هر کیلومتر از کابل مسی با هزار زوج تابیده نزدیک به ۸۰۰۰ کیلوگرم وزن

دارد، در حالیکه وزن کابل فیبر نوری با همان ظرفیت فقط ۱۰۰ کیلوگرم است. کاهش وزن نیز معادل است با کاهش نیاز به وسایل مکانیکی سنگین برای نصب و پشتیبانی سیستم، بگونه‌ایکه امروزه دیگر در مسیرهای جدید فقط از فیبر نوری استفاده می‌شود.

آخر اینکه، نور از فیبرهای نوری نشت نمی‌کند، و گرفتن انشعاب غیرمجاز از آن بسیار مشکل است. این ویژگی باعث شده تا فیبر نوری از ایمنی بسیار بالایی در مقابل سارقان اطلاعات برخوردار باشد.

البته فیبر نوری چندان بی‌عیب و نقص هم نیست. اول اینکه، این تکنولوژی هنوز بسیار جدید است و حتی بسیاری از مهندسین نیز توانایی کار با آن را ندارند. دیگر اینکه فیبر نوری بسیار آسیب‌پذیرتر از سیم مسی است، و حتی خم کردن بیش از حد باعث خرابی آن می‌شود. از آنجائیکه انتقال نوری ذاتاً یک‌طرفه است، برای ارتباط دوطرفه باید از دو رشته فیبر (یا یک رشته فیبر با دو باند فرکانسی) استفاده کرد. دست آخر اینکه، تجهیزات ارتباطی نوری گرانتر از انواع الکتریکی آن است. با این وجود، آیندهٔ مخابرات در فواصلی حتی بیش از چند متر از آنِ فیبرهای نوری است. برای بحثی جامع دربارهٔ جنبه‌های مختلف فیبر نوری و شبکه‌های آن، به (Hecht, 2001) مراجعه کنید.

۳ـ۲ انتقال بیسیم

یکی از پدیده‌های عصر ما **معتادان اینترنتی** است: کسانی که می‌خواهد بیست و چهار ساعته **بر-خط** (on-line) باشند. برای این قبیل افراد (که دائماً در حال جابجا شدن هستند) دیگر زوج تابیده، کابل کواکس و فیبر نوری کاربرد ندارد. آنها می‌خواهند بدون مقید شدن به هیچیک از سیستمهای مخابراتی دست و پا گیر، اطلاعات موردنیازشان را روی کامپیوتر کیفی، جیبی و حتی مچی(!) دریافت کنند؛ و مخابرات بیسیم تنها چیزیست که می‌تواند توقعات اینها را برآورده کند. در این قسمت نگاهی کلی به مخابرات بیسیم خواهیم انداخت، نه فقط بخاطر اینکه عده‌ای دوست دارند کنار دریا هم از دوستان اینترنتی‌شان جدا نشوند، بلکه به دلیل آنکه این تکنولوژی نقش مهمی در زندگی امروزی ما بازی می‌کند.

برخی افراد معتقدند که در آینده فقط دو نوع مخابرات وجود خواهد داشت: فیبر نوری و بیسیم. تمام تجهیزات ثابت (کامپیوترهای رومیزی، تلفن و فکس) از فیبر نوری، و تجهیزات متحرک از بیسیم بهره خواهند گرفت. حتی در مواردی می‌توان از بیسیم برای تجهیزات ثابت استفاده کرد. مثلاً، اگر کابل‌کشی به یک ساختمان (بعلت وجود موانع طبیعی مانند کوه، جنگل و باتلاق) مشکل باشد، استفاده از بیسیم ترجیح دارد. همانطور که قبلاً هم گفتیم، مخابرات بیسیم دیجیتال برای اولین بار در مجمع‌الجزایر هاوایی بکار گرفته شد.

۱ـ۳ـ۲ طیف الکترومغناطیس

امواج الکترومغناطیس حاصل حرکت الکترونها هستند، که می‌توانند در فضا (حتی در خلأ) منتشر شوند. وجود چنین امواجی اولین بار بصورت تئوری در سال ۱۸۶۵ توسط فیزیکدان انگلیسی جیمز کلرک ماکسول پیش‌بینی شد، و در سال ۱۸۸۷ فیزیکدان آلمانی هاینریش هرتز موفق شد آنها را مشاهده کند. تعداد نوسانهای یک موج در ثانیه **فرکانس**، f، نامیده می‌شود، و واحد آن (به افتخار هاینریش هرتز) **هرتز** (Hz) نامگذاری شده است. فاصلهٔ بین دو قله (یا حضیض) متوالی موج را نیز **طول موج** می‌گویند، و آنرا با حرف یونانی λ نشان می‌دهند.

اگر آنتنی با اندازهٔ مناسب به یک مدار الکتریکی وصل شود، می‌تواند امواج الکترومغناطیسی منتشر کند، که این امواج را می‌توان در فواصل مناسب دریافت کرد. تمام سیستمهای مخابرات بیسیم بر این اساس کار می‌کنند.

امواج الکترومغناطیسی (صرفنظر از فرکانس آنها) در خلأ با سرعت ثابتی (سرعت نور) حرکت می‌کنند، که مقدار تقریبی آن 3×10^8 m/sec است، و با c نشان داده می‌شود (سرعت نور سرعت حد در طبیعت است، یعنی

هیچ چیز نمی‌تواند سریعتر از آن حرکت کند.) سرعت این امواج در سیم مسی یا فیبر نوری به حدود $\frac{2}{3}c$ کاهش می‌یابد، که تا حدی نیز به فرکانس وابسته است.

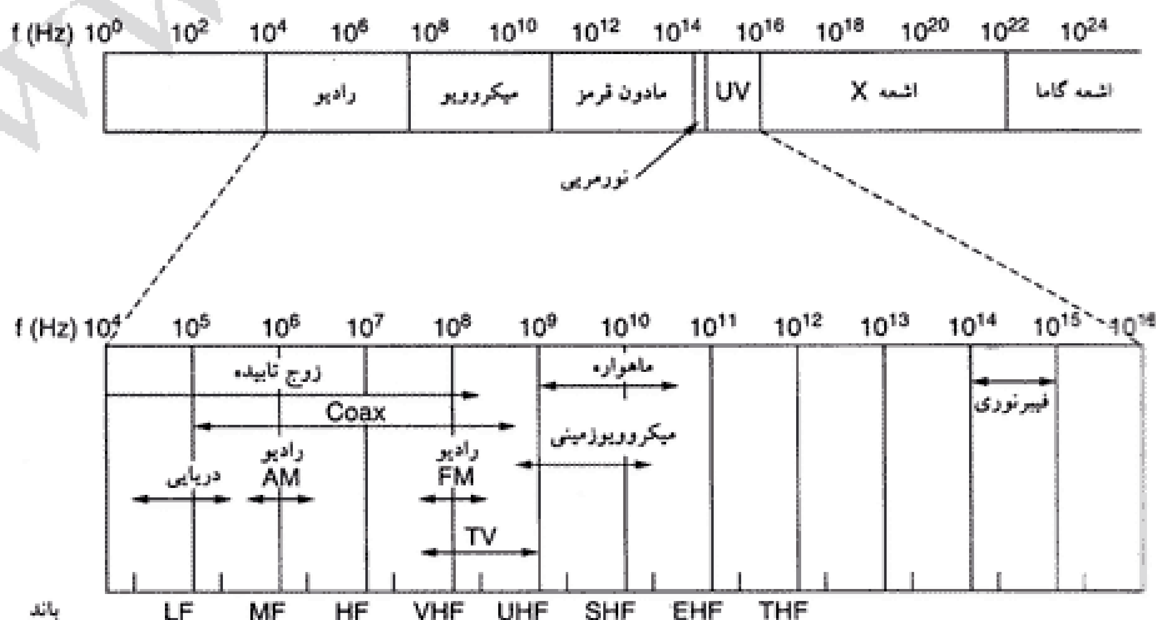
رابطهٔ اساسی بین f، λ و c (در خلأ) چنین است:

$$\lambda f = c \quad (۲-۲)$$

از آنجائیکه c ثابت است، با داشتن f می‌توان λ را محاسبه کرد، و بالعکس. اگر λ را برحسب متر و **f** را برحسب MHz داشته باشیم، رابطهٔ بالا بصورت $\lambda f \approx 300$ در می‌آید. برای مثال، در فرکانس 100-MHz طول موج 3 m است، که در فرکانس 1000-MHz طول موج آن به 0.3 m کاهش می‌یابد؛ موجی با طول موج 0.1 m فرکانسی معادل 3000-MHz (در خلأ) دارد.

در شکل ۲-۱۱ طیف الکترومغناطیس را ملاحظه می‌کنید. از ناحیه‌های رادیویی، مایکروبو، مادون‌قرمز، و مرئی این طیف می‌توان (با مدولاسیون دامنه، فرکانس، و فاز) برای انتقال اطلاعات استفاده کرد. نور ماوراءبنفش، اشعهٔ X و اشعهٔ گاما بدلیل فرکانس بالاتر برای منظور بهتر هستند، ولی از آنجائیکه تولید و مدولاسیون آنها مشکل است، از اجسام غیرشفاف عبور نمی‌کنند، و برای سلامتی موجودات زنده خطرناکند، کاربرد مخابراتی ندارند. باندهایی که در پائین شکل ۲-۱۱ می‌بینید، رسماً توسط ITU (برحسب طول موج) تقسیم‌بندی و نامگذاری شده‌اند. اصطلاحات LF، MF و HF بترتیب معادل **فرکانس پائین، فرکانس متوسط** و **فرکانس بالا** هستند. پیداست وقتی این نامگذاریها انجام می‌شد، کسی تصور فرکانسهای بالاتر از 10 MHz را هم نمی‌کرد، و بهمین دلیل امروزه برای باندهای بالاتر از آن از نامهای VHF (**فرکانس خیلی بالا**)، UHF (**فرکانس بسیار بالا**)، SHF (**فرکانس فوق‌العاده بالا**)، EHF (**فرکانس شدیداً بالا**)، و THF (**فرکانس خارق‌العاده بالا**) استفاده می‌شود - خوشبختانه برای فرکانسهای بالاتر هنوز نامی انتخاب نشده، و شما هم می‌توانید ذوق خود را در این زمینه امتحان کنید.

مقدار اطلاعاتی که یک موج الکترومغناطیس می‌تواند حمل کند، به پهنای باند آن بستگی دارد. با تکنولوژی امروزی، می‌توان در فرکانسهای پائین بازای هر هرتز از پهنای باند ۲ تا ۳ بیت، و در فرکانسهای بالا تا ۸ بیت



شکل ۲-۱۱. طیف الکترومغناطیس و کاربردهای مخابراتی آن.

اطلاعات منتقل کرد، بنابراین یک کابل کواکسیال با پهنای باند 750 MHz می‌تواند در هر ثانیه تا چندین گیگابیت اطلاعات منتقل کند. با یک نگاه به شکل ۲-۱۱ می‌توان براحتی دریافت که چرا فیبر نوری این همه طرفدار دارد.

اگر معادلهٔ (2-2) را برای f حل کرده و سپس از آن نسبت به λ دیفرانسیل بگیریم، خواهیم داشت

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2}$$

حال اگر بجای دیفرانسیل تفاضل محدود و قدر مطلق مقادیر را در نظر بگیریم، داریم

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2}$$

بنابراین با داشتن پهنای یک باند، $\Delta\lambda$، می‌توان فرکانس متناظر با آن باند، Δf، و از آنجا نرخ دادهٔ آن، را محاسبه کرد. هر چه پهنای یک باند بزرگتر باشد، نرخ انتقال دادهٔ آن باند بیشتر است. برای مثال، باند 1.30 μm را در شکل ۲-۶ در نظر بگیرید. در اینجا، $\lambda = 1.3 \times 10^{-6}$ m و $\Delta\lambda = 0.17 \times 10^{-6}$ m است، بنابراین Δf تقریباً 30 THz خواهد شد. با چنین پهنای باندی (و با فرض 8 bits/Hz) می‌توان به نرخ انتقال دادهٔ 240 Tbps دست یافت.

در اکثر سیستمهای انتقال از باندهای فرکانس باریک استفاده می‌شود (یعنی، $\frac{\Delta f}{f} \ll 1$)، تا گیرنده بتواند بیشترین توان را دریافت کند. اما گاهی نیز استفاده از باند فرکانسی وسیع ضروریست، که بر دو نوع است. در نوع **طیف گسترده با پرش فرکانسی** (frequency hopping spread spectrum)، فرستنده در هر ثانیه صدها بار فرکانس خود را عوض می‌کند (بعبارت دیگر از فرکانسی به فرکانس دیگر می‌پرد). این تکنیک بویژه در مخابرات نظامی کاربرد دارد، چون تشخیص فرکانس فرستنده بسیار مشکل است و پارازیت انداختن روی آن نیز تقریباً غیرممکن است. در این روش تضعیف موج در اثر انعکاس نیز وجود ندارد، چون وقتی موج انعکاسی (کمی بعد از موج اصلی) به گیرنده می‌رسد، فرکانس آن تغییر کرده است و فرکانسهای قبلی را دیگر قبول نمی‌کند. در سالهای اخیر این تکنیک در محصولات تجاری هم کاربرد پیدا کرده است - برای مثال، بلوتوث و 802.11 هر دو از این تکنیک استفاده می‌کنند.

اختراع این تکنیک نیز داستانی جالب دارد، که شنیدن آن خالی از لطف نیست. یکی از دو مخترع تکنیک طیف گسترده با پرش فرکانسی، خانم هدی لامار هنرپیشهٔ اتریشی‌الاصل است. شوهر اول این خانم، که سازندهٔ تسلیحات نظامی بود، برای وی توضیح داد که چگونه می‌توان با استفاده از سیگنالهای رادیویی اژدرها و موشک‌ها را کنترل کرد. وقتی خانم لامار فهمید که شوهرش به هیتلر اسلحه می‌فروشد، دچار وحشت شد، با لباس مبدل گریخت، و به هالیوود رفت تا به حرفهٔ موردعلاقه‌اش یعنی هنرپیشگی ادامه دهد. لامار به کمک دوستش جورج آنتیل (که یک آهنگساز بود) و برای کمک به متفقین، تکنیک پرش فرکانسی را اختراع کرد. طرح اولیهٔ آنها دارای ۸۸ فرکانس (به تعداد کلیدهای پیانو) بود، که به شمارهٔ 2,292,387 در ادارهٔ ثبت اختراعات ایالات متحدهٔ آمریکا به ثبت رسید. با این حال، آنها نتوانستند مفید و عملی بودن این اختراع را به نیروی دریایی آمریکا بقبولانند، و هرگز پولی بابت آن دریافت نکردند. تنها سالها پس از سپری شدن از حق‌الاختراع آن بود که این تکنیک مورد توجه محافل علمی قرار گرفت.

نوع دوم باندهای فرکانسی وسیع، **طیف گسترده با توالی مستقیم** (direct sequence spread spectrum) نام دارد، که در آن سیگنال روی طیف وسیعی از فرکانسها پخش می‌شود. این تکنیک نیز امروزه، بویژه در تلفنهای همراه نسل دوم، کاربردهای تجاری پیدا کرده است، و با آمدن نسل سوم تلفنهای همراه تسلط آن بر بازار کامل خواهد شد، زیرا دارای کارایی طیفی خوب، مصونیت در برابر نویز عالی و بسیاری ویژگیهای دیگر است. در برخی از شبکه‌های محلی بیسیم نیز از تکنیک طیف گسترده با توالی مستقیم استفاده شده است. در قسمتهای آیندهٔ همین

فصل باز هم به مبحث طیف گسترده برمی‌گردیم؛ اگر به تکنیکهای مخابرات طیف گسترده و تاریخچهٔ آن علاقمند هستید، کتاب (Scholtz, 1982) را جالب خواهید یافت.

فعلاً فرض را بر این می‌گذاریم که همه جا از باند فرکانسی باریک استفاده می‌شود، و بحث خود را با کاربرد بخشهای مختلف طیف الکترومغناطیس شکل ۲-۱۱ ادامه می‌دهیم - از رادیو شروع می‌کنیم.

۲-۳-۲ مخابرات رادیویی

امواج رادیویی کاربرد گسترده‌ای در مخابرات (در فضاهای سرپوشیده یا باز) دارند، چون بآسانی می‌توان آنها را تولید کرد، بُرد زیادی دارند، و از ساختمانها و موانع عبور می‌کنند. امواج رادیویی همه-طرفه هستند، یعنی در تمام جهات منتشر می‌شوند، بنابراین نیازی به تنظیم دقیق موقعیت گیرنده و فرستنده نسبت بیکدیگر نیست.

البته همه-طرفه بودن امواج رادیویی همیشه هم خوب نیست. در دههٔ ۱۹۷۰، شرکت جنرال موتورز تصمیم گرفت اتومبیلهای کادیلاک جدید خود را به ترمز ضد-قفل کامپیوتری (ABS) مجهز کند. در این سیستم وقتی راننده ترمز می‌گیرد، کامپیوتر مرکزی پالسهایی به ترمزها می‌فرستد، که آنها را چندین بار در ثانیه باز و بسته می‌کند، و ترمزها دیگر قفل نمی‌کنند. مدتی بعد در یک روز آفتابی، یکی از پلیسهای گشت بزرگراه ایالت اوهایو تصمیم گرفت رادیوی بیسیم جدید خود را امتحان کرده و با مرکز فرماندهی تماس بگیرد. با این کار، اتومبیل کادیلاکی که در نزدیکی وی حرکت می‌کرد، بلافاصله مثل یک اسب چموش شروع به حرکات عجیبی کرد. وقتی افسر پلیس اتومبیل خاطی را متوقف کرد، راننده ادعا کرد که هیچ کاری انجام نداده و اتومبیل بیکباره دیوانه شده است.

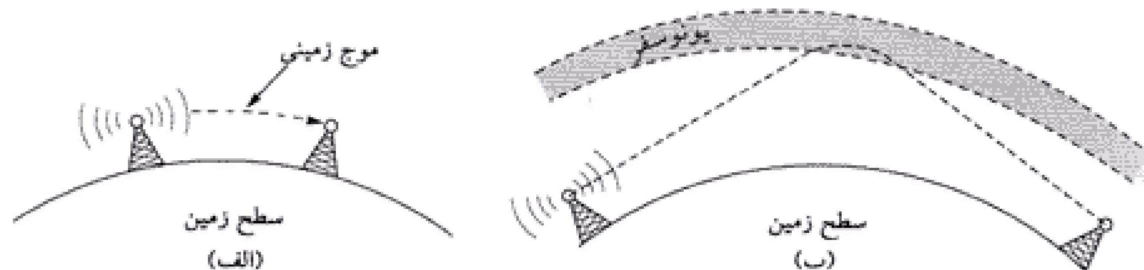
کمی بعد ماجرا روشنتر شد: اتومبیلهای کادیلاک همه جا خوب کار می‌کردند، و فقط در بزرگراههای ایالت اوهایو، آن هم وقتی پلیس می‌دیدند، دیوانه می‌شدند. تا مدتها جنرال موتورز سرگردان بود و نمی‌توانست بفهمد چرا کادیلاک در همه جا جز بزرگراههای ایالت اوهایو خوب کار می‌کند (کادیلاک حتی در خیابانها و جاده‌های معمولی اوهایو هم خوب کار می‌کرد). فقط بعد از تحقیقات گسترده بود که معلوم شد، مدارهای سیمکشی کادیلاک برای فرکانس سیستم رادیویی جدید پلیس بزرگراه اوهایو تبدیل به یک آنتن خوب می‌شود، و پالسهای این رادیو سیستم ABS را فعال می‌کند.

ویژگیهای امواج رادیویی به فرکانس آنها وابسته است. در فرکانسهای پائین، امواج براحتی از موانع عبور می‌کنند، ولی توان آنها بر حسب فاصله بسرعت افت می‌کند (با ضریب $1/r^2$). در فرکانسهای بالا، امواج رادیویی به خط مستقیم حرکت می‌کنند، و در برخورد با موانع منعکس می‌شوند. حتی قطرات باران هم امواج با فرکانس بالا را جذب می‌کند. امواج رادیویی در تمام فرکانسها در معرض تداخل ناشی از کارکرد وسایل الکتریکی (مانند موتور) هستند.

بدلیل بُرد زیاد امواج رادیویی، خطر تداخل آنها در سرتاسر دنیا وجود دارد، بهمین دلیل دولتها نحوهٔ استفاده از فرستنده‌های رادیویی را (البته با یک استثنا) بشدت کنترل می‌کنند.

امواج رادیویی در باندهای VLF ، LF و MF از انحنای زمین تبعیت می‌کنند - به شکل ۲-۱۲ (الف) نگاه کنید. این امواج نزدیک به ۱۰۰۰ کیلومتر بُرد دارند. ایستگاههای رادیویی AM در باند MF کار می‌کنند، و بهمین دلیل بُرد آنها چنین زیاد است. امواج رادیویی در این باند براحتی از موانع و ساختمانها عبور می‌کنند، و علت دریافت آنها در داخل ساختمانها نیز همین است. تنها مشکل این امواج پهنای باند کم آنهاست (معادلهٔ 3-2 را ببینید).

در باندهای HF و VHF ، آن قسمت از امواج رادیویی که نزدیک سطح زمین حرکت می‌کنند، جذب زمین می‌شوند. ولی بخشی از امواج که به سمت فضا می‌روند، پس از برخورد با **یونوسفر** (ionosphere - لایه‌ای از جو زمین به ارتفاع ۱۰۰ تا ۵۰۰ کیلومتر، که حاوی ذرات باردار است) به طرف زمین برمی‌گردند؛ شکل ۲-۱۲ (ب) را



شکل ۲-۱۲. (الف) در باندهای VLF ، LF و MF امواج رادیویی از انحنای زمین تبعیت می‌کنند. (ب) در باندهای HF و VHF آنها بین زمین و یونوسفر رفت و برگشت می‌کنند.

نگاه کنید. در شرایط خاصی این امواج می‌توانند چندین بار بین زمین و یونوسفر رفت و برگشت کرده، و صدها کیلومتر دورتر دریافت شوند. طرفداران رادیو آماتوری و مخابرات نظامی از باندهای HF و VHF استفاده می‌کنند.

۲-۳-۳ مخابرات مایکروویو

در فرکانسهای بالای 100 MHz امواج تقریباً به خط مستقیم حرکت می‌کنند، و می‌توان آنها را دقیقاً روی یک نقطه متمرکز کرد. متمرکز کردن تمام انرژی موج در یک پرتو باریک (با استفاده از آنتنهای بشقابی) باعث بالا رفتن نسبت سیگنال به نویز می‌شود، ولی از طرف دیگر لازم است تا گیرنده و فرستنده بدقت تنظیم شوند. علاوه بر آن، باریک بودن پرتوها باعث می‌شود تا تداخل فرستنده‌ها به حداقل برسد، مشروط بر اینکه کمی با هم فاصله داشته باشند. تا قبل از اختراع فیبر نوری، این امواج (که به مایکروویو معروفند) ستون فقرات مخابرات راه دور محسوب می‌شدند. در واقع، یکی از شرکتهای رقیب AT&T بنام MCI شبکهٔ گسترده‌ای از برجهای مایکروویو با فواصل دهها کیلومتر ایجاد کرده بود (حتی نام این شرکت – Microwave Communications, Inc. – هم بنوعی گویای استفاده از امواج مایکروویو است). MCI مدتهاست به فیبر نوری روی آورده، و اکنون در WorldCom ادغام شده است.

از آنجائیکه امواج مایکروویو به خط مستقیم حرکت می‌کنند، اگر فاصلهٔ ایستگاه مبدأ و مقصد زیاد باشد، انحنای زمین مانع از رسیدن امواج خواهد شد. به همین دلیل لازم است تا در فاصلهٔ بین آنها از برجهای تکرارکننده استفاده شود. هر چه ارتفاع این برجها بیشتر باشد، فاصلهٔ آنها می‌تواند زیادتر باشد. فاصلهٔ دو ایستگاه تقریباً با توان دوم ارتفاع آنها رابطه دارد؛ برای مثال، دو برج ۱۰۰ متری می‌توانند چیزی در حدود ۸۰ کیلومتر فاصله داشته باشد.

امواج مایکروویو، برخلاف امواج رادیویی فرکانس پائین، نمی‌توانند بخوبی از موانع عبور کنند. علاوه بر این، با وجود متمرکز شدن موج در فرستنده، امواج مایکروویو در طول مسیر دچار پراکندگی جزیی می‌شوند. قسمتی از موج مایکروویو نیز که توسط لایه‌های پائین جو منعکس می‌شود، فاصلهٔ بیشتری را طی کرده و هنگام رسیدن به گیرنده دیگر با موج اصلی هم فاز نیست، و باعث خنثی شدن آن می‌شود. این پدیده که به **محوشدگی چندمسیره** (multipath fading) معروف است، یکی از مشکلات جدی در مخابرات رادیویی محسوب می‌شود، و به وضع هوا و فرکانس موج بستگی دارد. در برخی از موارد، ۱۰ درصد ظرفیت کانال برای مقابله با این وضعیت کنار گذاشته می‌شود، تا در صورت بروز محوشدگی چندمسیره بتوانند موقتاً از فرکانسهای جایگزین استفاده کنند.

تقاضا برای پهنای باند بیشتر باعث شده تا فرکانسها هر روز بالا و بالاتر برود. امروزه باندهای 10 GHz نیز در حال کار هستند، ولی از فرکانس 4 GHz به بالا مشکل جدیدی خودنمایی می‌کند: جذب شدن انرژی امواج توسط

آب. طول موج این قبیل امواج فقط چند سانتی‌متر است، و براحتی جذب قطرات باران می‌شوند. این پدیده شاید برای کسانی که می‌خواهند یک اجاق مایکروویو بزرگ بسازند و پرندگان را در هوا کباب کنند خوب باشد، ولی برای مخابرات مایکروویو یک مشکل جدی است. تنها راه حل این مشکل (مانند محوشدگی چندمسیره) قطع کردن کانالهای باران‌زده، و استفاده از کانالهای جایگزین است.

خلاصه اینکه، استفاده از امواج مایکروویو در مخابرات راه‌دور، تلفنهای همراه، و تلویزیون چنان گسترش یافته، که امروزه دیگر در طیف مایکروویو جایی برای استفاده بیشتر نمانده است. مخابرات مایکروویو مزایای زیادی نسبت به فیبر نوری دارد. اول اینکه نیازی به تصرف زمین برای حفر کانال و کشیدن کابل نیست، و با خرید چند تکه زمین کوچک در فواصل ۵۰ کیلومتری و نصب چند برج مایکروویو، می‌توان یک سیستم مخابراتی مستقل ایجاد کرد. بهمین خاطر بود که MCI توانست بسرعت بعنوان یکی از غولهای صنعت تلفن راه‌دور قد علم کند. (یکی دیگر از شرکتهای بزرگ مخابراتی یعنی Sprint ، مسیر کاملاً متفاوتی در پیش گرفت: این شرکت که متعلق به **راه‌آهن پاسیفیک جنوبی** بود، از زمینهای حاشیهٔ خطوط آهن استفاده کرده، وکابلهای خود را در آنجا دفن کرد.)

دیگر اینکه، مایکروویو نسبتاً ارزان است. نصب چند برج ساده برای نصب آنتنها (که می‌تواند فقط یک دکل ساده با چهار سیم مهارکننده باشد)، بسیار ارزانتر از خرید زمین در مناطق شلوغ شهری یا صعب‌العبور کوهستانی (و یا حتی اجارهٔ خطوط تلفنی از شرکتهای تلفن) است.

طیف الکترومغناطیس و سیاست

برای جلوگیری از هرج و مرج و آشوب، توافقهایی در سطح ملی و بین‌المللی برای نحوهٔ استفاده از فرکانسها شکل گرفته است. از آنجائیکه هر کس پهنای باند بیشتری می‌خواهد، دولتها مجبور به دخالت هستند، و پهنای باند لازم برای رادیوی AM و FM ، تلویزیون، تلفنهای همراه، شرکتهای تلفن، پلیس، ناوبری هوایی و دریایی، مصارف نظامی و دولتی (و خلاصه، همهٔ آنهایی که فرکانس می‌خواهند) را به آنها اختصاص می‌دهند. در سطح بین‌المللی نیز یکی از شعبات ITU-R (بنام WARC) به هماهنگی این اقدامات اختصاص یافته، تا کارکرد وسایل مخابراتی را در کشورهای مختلف تضمین کند. با این همه، توصیه‌های ITU-R الزام‌آور نیست، و گاهی پیش می‌آید که FCC (Federal Communication Commission – سازمانی که مسئول تخصیص باندهای فرکانسی در ایالات متحدهٔ آمریکاست) توصیه‌های آنرا نادیده بگیرد، تا (مثلاً) یک طیف فرکانسی را به یک گروه قدرتمند سیاسی اختصاص دهد.

علاوه بر اختصاص قسمتی از طیف فرکانسی به یک کاربرد خاص (مثلاً، تلفن همراه)، باید مشخص شود که کدام **حامل** (carrier) ها مجاز به استفاده از این فرکانسها هستند. در گذشته، برای این منظور سه روش کاربرد گسترده‌تری داشت. در قدیمی‌ترین آنها، که به **مسابقهٔ زیبایی** معروف بود، هر حامل بایستی توضیح می‌داد که چرا پیشنهاد آنها بیشترین نفع را برای جامعه دارد - و این مقامات دولتی بودند که بهترین پیشنهاد را انتخاب می‌کردند. از آنجائیکه انتخاب یک حامل منافع میلیاردی برای آن در بر داشت، در این روش رشوه، فساد و پارتی‌بازی بیداد می‌کرد. علاوه بر آن، حتی وقتی یک کارمند وظیفه‌شناس تشخیص می‌داد که پیشنهاد یک شرکت خارجی بهتر از شرکتهای داخلی است، قبولاندن آن به مقامات بالاتر کار ساده‌ای نبود.

مشکلات روش انتخاب دولتی، به روش دوم منجر شد: قرعه‌کشی بین حامل‌های متقاضی یک طیف فرکانسی. مشکل این روش آن بود که حتی شرکتهایی که هیچ نفعی در استفاده از طیف نداشتند، می‌توانستند در قرعه‌کشی شرکت کنند. بدین ترتیب حتی صاحب یک رستوران یا کفش فروشی هم می‌توانست برندهٔ قرعه‌کشی شده، و بعد با خیال راحت و بدون هیچ خطری امتیاز خود را به شرکتهای ذینفع بفروشد و سود کلانی به جیب بزند.

بخشیدن چنین ثروتهای بادآورده‌ای به افراد زرنگ و هوشیار، باعث انتقادات زیادی شد، و منجر بـه اتـخاذ روش سوم اعطای امتیاز حامل طیفهای فرکانسی گردید: حراج کردن پهنای باند. وقتی انگلستان در سـال ۲۰۰۰ طیف فرکانسی تلفنهای همراه نسل سوم را به حراج گذاشت، پیش‌بینی می‌کرد که از ایـن راه چـهار میلیارد دلار عایدی داشته باشد. ولی در واقع از این مزایده چهل میلیارد دلار سود برد، چون شرکتهای زیادی (از ترس اینکه از قافلهٔ تلفن همراه عقب بمانند) به تکاپوی خرید طیف فرکانسی افتادند. این حادثه طمع سایر دولتها را تـحریک کرد، و آنها هم به فکر راه انداختن مزایدهٔ فروش طیف فرکانسی افتادند. شرکتهای بسیاری به همین دلیل تا مرز ورشکستگی پیش رفتند، و آنهایی که ماندند باید سالها تلاش کنند تا پول از دست رفته را جبران کنند.

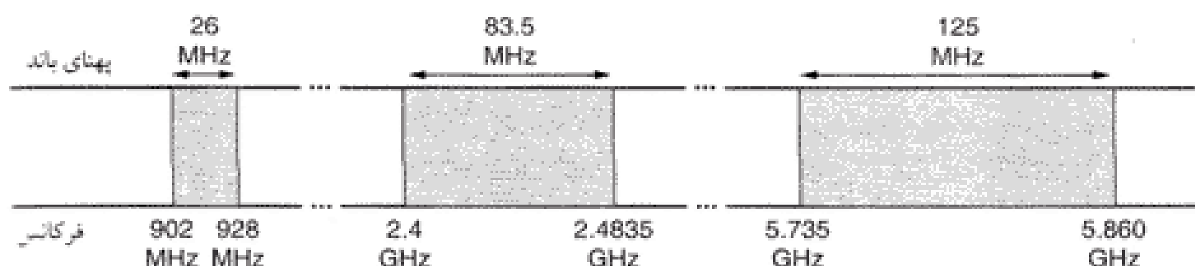
روش دیگر تخصیص فرکانس این است که اصلاً آنرا به کسی اختصاص ندهند: اجازه دهید هر کس بـا هـر فرکانسی که می‌خواهد کار کند، فقط توان فرستنده‌های آنها را بگونه‌ای کنترل کنید که بُرد کمی داشته باشند، و با فرستنده‌های دیگر تداخل نکنند. بسیاری از دولتها بخشی از طیف فرکانسی خود را (که به باند ISM : **صنعتی، علمی، پزشکی** معروف است) با همین نیت آزاد می‌گذارند تا هر کس که مایل است از آنها استفاده کند. در بـاز کن‌های برقی، تلفنهای بیسیم خانگی، اسباب‌بازی‌های کنترل از راه دور، و بسیاری از وسایل خانگی مجهز به کنترل رادیویی از باندهای ISM استفاده می‌کنند. برای به حداقل رساندن تداخل بین این دستگاهها، FCC و سایر دولتها سازندگان این قبیل وسایل را مجبور می‌کنند تا از تکنیکهای طیف گسترده استفاده کنند.

محل طیف ISM از کشوری به کشور دیگر متفاوت است. برای مثال، در ایالات متحدهٔ آمریکا وسایلی با توان تشعشعی کمتر از ۱ وات می‌توانند (بدون نیاز به اخذ مجوز FCC) از باندهای مشخص شـده در شـکل ۲-۱۳ استفاده کنند. باند 900-MHz بهترین کارایی را دارد، ولی علاوه بر اینکه خیلی شلوغ است، در تمام دنیا هم آزاد نیست. باند 2.4-GHz در اغلب کشورهای دنیا آزاد است، ولی در معرض تداخل امـواج اجـاقهای مـایکروویو و تأسیسات رادار قرار دارد. بلوتوث و برخی از شبکه‌های محلی 802.11 در این باند کار می‌کنند. بـاند 5.7-GHz نسبتاً جدید است و وسایل آن هنوز گران هستند، ولی از آنجائیکه استاندارد 802.11a در این باند کـار مـی‌کند، بزودی شاهد رواج آن خواهیم بود.

۲-۳-٤ امواج مادون‌قرمز و میلیمتری

امواج هدایت‌نشدهٔ مادون‌قرمز و میلیمتری کاربرد زیادی در مخابرات بُرد کوتاه دارند. دستگاههای کـنترل از راه دور وسایل صوتی و تصویری همگی از امواج مادون‌قرمز استفاده می‌کنند. این دستگاهها جهت‌دار، ارزان و ساده هستند، ولی یک عیب عمده دارند: امواج مادون‌قرمز از اجسام صلب عبور نمی‌کند (بین دستگاه کنترل از راه دور و تلویزیون خود بایستید، و ببینید هنوز کار می‌کند یا نه). در حالت کلی، هر چه به سمت فرکانسهای طیف مرئی نور برویم، امواج بیشتر شبیه نور (و کمتر شبیه امواج رادیویی) عمل می‌کنند.

از طرف دیگر، عبور نکردن امواج مـادون‌قرمز از مـوانـع سـخت یـک مـزیت مـحسوب مـی‌شود: شـما کـه نمی‌خواهید تلویزیونتان با کنترل از راه دور همسایه روشن و خاموش شود. استراق سمع امواج مادون‌قرمز نـیز



شکل ۲-۱۳. باندهای ISM در ایالات متحدهٔ آمریکا.

مشکلتر از امواج رادیویی است، و بهمین دلیل از ایمنی بالاتری برخوردارست. بر خلاف امواج رادیویی، استفاده ازامواج مادون‌قرمز هیچ نیازی به کسب مجوز از مقامات رسمی ندارد. با وجود برخی کاربردهای امواج مادون‌قرمز در صنعت کامپیوتر (مانند اتصال ماوس و چاپگر)، این امواج در صحنهٔ مخابرات حرفی برای گفتن ندارند.

۲ـ۳ـ۵ مخابرات امواج نوری

ارسال علائم نوری قرنهاست که شناخته شده و بکار برده می‌شود. امروزه برای متصل کردن دو شبکه که در ساختمانهای جداگانه قرار دارند، از لیزرهایی که روی پشت بام ساختمانها قرار می‌گیرند، استفاده می‌شود. پرتوهای لیزر اساساً یکطرفه هستند، بنابراین هر ساختمان باید فرستنده و گیرندهٔ لیزری جداگانه داشته باشد. پهنای باند این روش بسیار بالا و هزینهٔ آن نیز بسیار پائین است، و علاوه بر اینکه نصب ساده‌ای (نسبت به تجهیزات مایکروویو) دارد، نیازی به مجوز FCC نیز ندارد.

البته مزیت اصلی پرتو لیزر (یعنی باریک بودن آن) در اینجا یک نقطه ضعف محسوب می‌شود. هدف‌گیری یک پرتو لیزر به قطر ۱ میلی‌متر روی هدفی باندازهٔ ته سوزن در فاصلهٔ ۵۰۰ متری حتی برای قهرمان تیراندازی المپیک هم دشوار است. در این سیستمها معمولاً از عدسیهای خاصی برای پراکنده کردن نور لیزر و پهن کردن پرتوهای آن استفاده می‌شود.

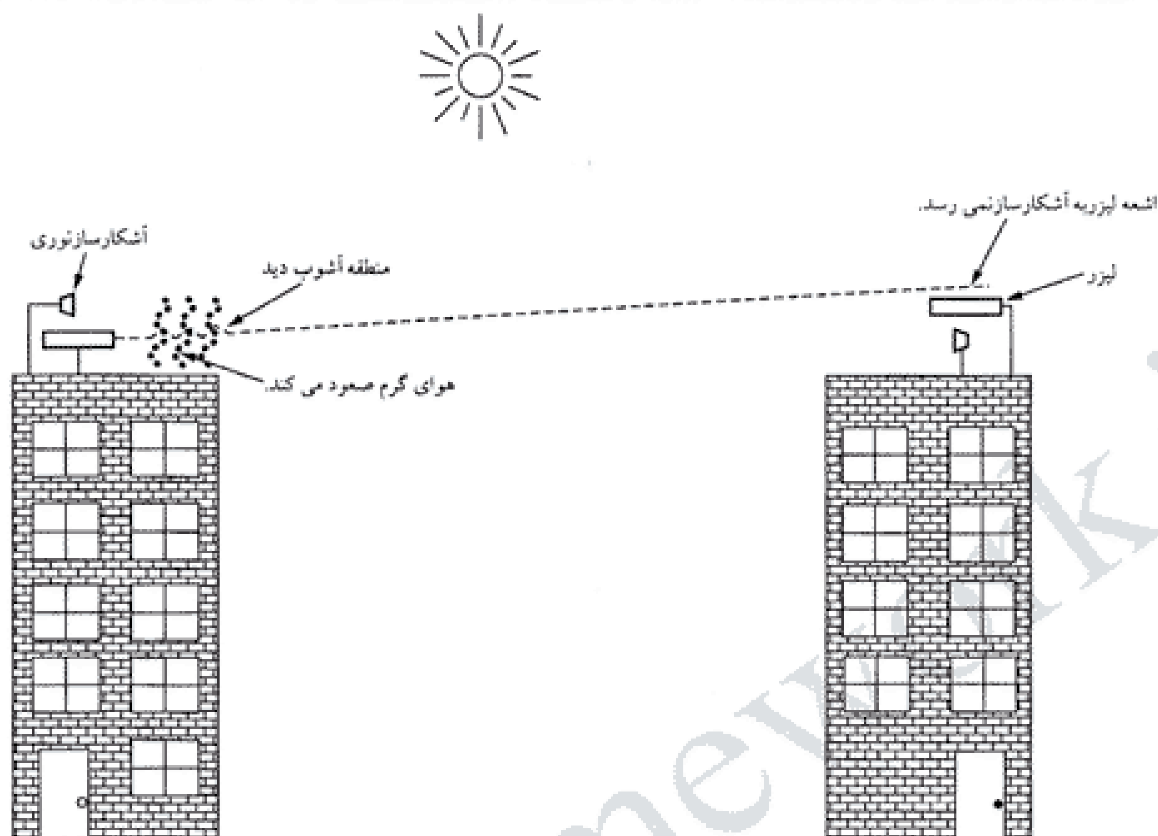
یکی دیگر از معایب لیزر اینست که نمی‌تواند در روزهای بارانی و مه‌آلود بخوبی عمل کند، ولی برای هوای صاف و آفتابی ایده‌آل است. تجربهٔ جالبی که مؤلف این کتاب از چنین سیستمی دارد، می‌تواند آموزنده باشد. چندی قبل در یک کنفرانس علمی که در یکی از هتلهای مدرن اروپایی ترتیب یافته بود، شرکت داشتم. مسئولین با درایت کنفرانس تعدادی کامپیوتر در یک اتاق نصب کرده بودند، تا مدعوین بتوانند در حین شرکت در سخنرانی‌ها ایمیل خود را نیز چک کنند. از آنجائیکه شرکت مخابرات محلی حاضر نشده بود تعداد زیادی خط تلفن را فقط برای سه روز برگزاری کنفرانس در اختیار برگزارکنندگان آن قرار دهد، آنها یک دستگاه لیزر روی پشت بام هتل قرار داده بودند، تا آنجا را به ساختمان دپارتمان کامپیوتر دانشگاه که در چند کیلومتری آن محل قرار داشت، وصل کنند. سیستم شب قبل از افتتاح کنفرانس تست شده بود، و همه چیز مرتب بنظر می‌رسید. ساعت ۹ صبح روز بعد (که روزی صاف و آفتابی بود) ارتباط بکلی قطع شد، و تمام آن روز نیز وصل نشد. عصر همان روز، برگزارکنندگان کنفرانس دوباره سیستم را تست کردند، و این بار هم همه چیز درست و عالی بود. اما صبح روز بعد (و روز بعد از آن) دوباره همان اتفاق تکرار شد. فقط بعد از پایان کنفرانس بود که علت کشف شد.

با شروع روز، گرمای خورشید باعث گرم شدن بام ساختمان شده، و هوای گرم به بالا صعود می‌کرد (شکل ۲-۱۴). این جریان هوای آشفته باعث می‌شد تا پرتو لیزر به مقدار ناچیز منحرف شده، و تمرکز آن روی گیرندهٔ ساختمان هدف بهم بخورد. این دقیقاً همان پدیده‌ایست که باعث می‌شود تا ستارگان در شب چشمک بزنند، و یا در روزهای داغ تابستان در جاده‌ها همه چیز مؤاج بنظر برسد.

۴ ماهواره‌های مخابراتی

در سالهای ۱۹۵۰ و اوایل ۱۹۶۰ برخی افراد تلاش کردند تا با استفاده از بالونهای هوای گرم که پوششی فلزی داشتند، نوعی سیستم انعکاس امواج رادیویی بسازند. متأسفانه سیگنال برگشتی چنان ضعیف بود که به هیچ دردی نمی‌خورد. سپس نیروی دریایی آمریکا متوجه بالونی شد که همیشه در آسمان است: ماه؛ سیستمی که ساخته شد با استفاده از انعکاس امواج از ماه، بین کشتی‌ها و پایگاههای ساحلی ارتباط برقرار می‌کرد.

پیشرفت بیشتر در مخابرات فضایی فقط زمانی ممکن شد که انسان اولین ماهواره‌های مخابراتی را به فضا



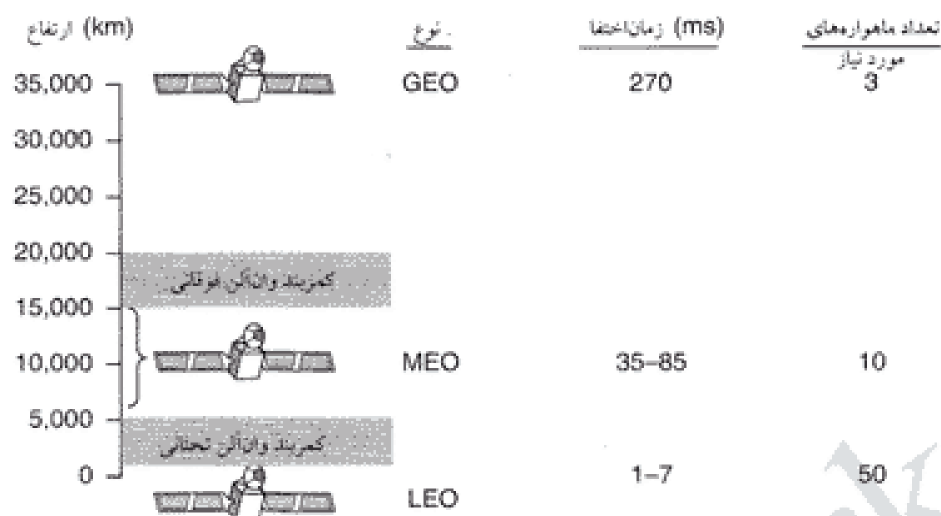
شکل ۲-۱۴. جریانهای همرفتی می‌تواند باعث انحراف پرتوهای لیزر شود.

پرتاب کرد. تفاوت کلیدی بین ماهواره‌های مصنوعی و اجسام فضایی این بود که ماهواره‌های ساختهٔ دست بشر می‌توانستند قبل از برگشت دادن سیگنال آنرا تقویت کنند - بدین ترتیب، یک کنجکاوی عجیب تبدیل به یکی از ارکان زندگی بشر شد.

ماهواره‌های مخابراتی چند ویژگی دارند که آنها را برای کاربردهای مختلف جذاب می‌کند. یک ماهوارهٔ مخابراتی، در ساده‌ترین شکل خود، یک تکرارکنندهٔ مایکروویو بزرگ در آسمان است. چنین ماهواره‌ای دارای چندین **ترانسپاندر** (تکرارکننده - transponder) است، که به فرکانسهای خاصی گوش می‌کنند، سیگنال دریافتی را تقویت کرده، و سپس آنرا با فرکانس متفاوتی برمی‌گردانند (تا با سیگنال ورودی تداخل نکند). پرتو برگشتی می‌تواند وسیع باشد و بخش بزرگی از سطح کرهٔ زمین را بپوشاند، یا اینکه باریک باشد و فقط منطقه‌ای به قطر چند صد کیلومتر مربع را پوشش دهد. به این حالت **شیپور خمیده** (bent pipe) گفته می‌شود.

طبق قانون کپلر، دورهٔ گردش مداری یک ماهواره با توان $\frac{۳}{۲}$ شعاع مدار آن متناسب است - هر چه مدار ماهواره بالاتر باشد، دورهٔ گردش آن بیشتر است. در مدارهای نزدیک زمین دورهٔ گردش ماهواره‌ها معمولاً ۹۰ دقیقه است، و آنها بسرعت از دید ایستگاههای زمینی خارج می‌شوند. برای پوشش دادن تمام سطح کرهٔ زمین با چنین ماهواره‌هایی، به تعداد زیادی از آنها نیاز است. در مداری به ارتفاع 35,800 km دورهٔ گردش ماهواره ۲۴ ساعت (معادل یک دور گردش زمین به دور خود) است. اگر ارتفاع ماهواره را تا 384,000 km بالا ببریم، دورهٔ گردش آن یک ماه خواهد شد - اگر باور ندارید، به ماه نگاه کنید؛ ارتفاع مداری ماه از سطح زمین دقیقاً همین مقدار است.

دورهٔ گردش ماهواره اهمیت زیادی دارد، ولی تنها عاملی نیست که تعیین می‌کند ماهواره کجا باید قرار گیرد. عامل مهم دیگر، وجود **کمربند وان آلن** (Van Allen belt) است - کمربند وان آلن لایه‌ای از ذرات باردار پُرانرژی است که توسط میدان مغناطیسی زمین بدام افتاده‌اند. هر ماهواره‌ای که در داخل این لایه پرواز کند.



شکل ۲-۱۵. ماهواره‌های مخابراتی و برخی از ویژگیهای آنها، از جمله ارتفاع از سطح زمین، زمان تأخیر رفت و برگشت سیگنال، و تعداد ماهواره‌هایی که برای پوشش کل زمین لازم است.

بسرعت توسط ذرات باردار پُرانرژی آن از بین می‌رود. از ترکیب این دو عامل سه ناحیه مشخص می‌شود، که می‌توان ماهواره‌ها را با اطمینان خاطر در آنها قرار داد (شکل ۲-۱۵ را ببینید). در قسمتهای آینده ماهواره‌هایی را که در هر یک از این ناحیه‌ها قرار داده می‌شوند، تشریح خواهیم کرد.

۲-٤-۱ ماهواره‌های زمین‌ثابت

در سال ۱۹۴۵، نویسندهٔ داستانهای علمی-تخیلی آرتور سی. کلارک با محاسبات خود نشان داد که اگر ماهواره‌ای در یک مدار استوایی و در ارتفاعی معادل 35,800 km قرار گیرد، نسبت به زمین ثابت بنظر خواهد رسید، و نیازی نیست که ایستگاه زمینی آنرا تعقیب کند (Clarke, 1945) . وی در این مقاله یک سیستم کامل مخابراتی را با استفاده از این **ماهواره‌های زمین‌ثابت** (geostationary satellites) تشریح کرده بود، و از جمله مدار آنها، پانلهای خورشیدی (برای تأمین انرژی موردنیاز)، فرکانسهای رادیویی، و نحوهٔ پرتاب ماهواره‌ها را بدقت توضیح داده بود. متأسفانه، وی در این مقاله نتیجه گرفته بود که این طرح غیرعملی است، چون قرار دادن تقویت‌کننده‌های لامپ خلأ (که بسیار پُرمصرف، حجیم و شکننده بودند) در مدار زمین غیرممکن است، و هرگز این ایده را دنبال نکرد، اگر چه چند داستان علمی-تخیلی در این زمینه نوشت.

اختراع ترانزیستور همه چیز را تغییر داد، و اولین ماهوارهٔ مخابراتی بنام **تل‌استار** (Telstar) در ژوئیهٔ ۱۹۶۲ به فضا پرتاب شد. از آن زمان به بعد، ماهواره‌های مخابراتی صنعتی چندمیلیارد دلاری را بوجود آورده‌اند، که تنها زمینهٔ سودآور تحقیقات فضایی است. به این ماهواره‌های بلندپرواز اغلب ماهواره‌های GEO (Geostationary Earth Orbit) گفته می‌شود.

با تکنولوژی موجود (برای جلوگیری از تداخل امواج) حداقل فاصلهٔ دو ماهوارهٔ زمین‌ثابت نمی‌تواند کمتر از ۲ درجه در صفحهٔ استوایی باشد. بدین ترتیب، در هر زمان بیش از $360 / 2 = 180$ نمی‌توانند در مدار باشند. با این حال برای بالا بردن پهنای باند در دسترس، می‌توان در هر ترانسپاندر از فرکانسها و پولاریزاسیونهای مختلف استفاده کرد.

در این مورد هم برای جلوگیری از هرج و مرج در مدار زمین، تخصیص مکانهای مداری توسط ITU انجام می‌شود. پای سیاست به اینجا هم باز شده است: کشورهایی که بزحمت از عصر حجر فاصله گرفته‌اند، سهم خود

را از مکانهای مداری طلب می‌کنند، تا بعد بتوانند آنرا به قیمت خوب به متقاضیان اجاره بدهند. کشورهایی هم هستند که ادعا می‌کنند قلمرو خاک آنها تا کرهٔ ماه ادامه دارد، و هیچکس حق ندارد در آسمان بالای سر آنها ماهواره داشته باشد. و برای شلوغی بیشتر اوضاع، فقط مخابراتی‌ها نیستند که بر سر تصاحب مدارها می‌جنگند: تلویزیونهای ماهواره‌ای، دولتها، و نظامیها هم می‌خواهند سهمی از مدارهای زمین داشته باشند.

ماهواره‌های امروزی بسیار بزرگ هستند، که وزن آنها گاهی به ۴۰۰۰ کیلوگرم می‌رسد، و دهها کیلووات انرژی الکتریکی مصرف می‌کنند (انرژیی که توسط پانلهای خورشیدی تولید می‌شود). مدار و موقعیت این ماهواره‌ها تحت تأثیر جاذبهٔ خورشید، ماه و سایر سیارات منظومهٔ شمسی پیوسته در حال تغییر است، که این تأثیرات توسط موتورهای موشکی کوچکی که در آنها تعبیه شده، خنثی می‌شود (به این کار **نگهداری ایستگاه** – station keeping – می‌گویند). از آنجائیکه سوخت این موتورها محدود است، بعد از مدتی (که معمولاً حدود ۱۰ سال است) ماهواره بکلی از کنترل خارج و بلااستفاده می‌شود. ارتفاع چنین ماهواره‌هایی بتدریج کاهش می‌یابد، و پس از ورود به جو زمین می‌سوزند، و یا در برخورد با سطح زمین متلاشی می‌شوند.

مکانهای مداری تنها چیزی نیست که سر آن دعواست. فرکانسها نیز خواهان زیادی دارد، چون احتمال تداخل فرکانسهای ارسالی از ماهواره (downlink) با فرکانسهای زمینی وجود دارد. برای رفع این مشکل، ITU چند باند فرکانسی را به کاربردهای ماهواره‌ای اختصاص داده است، که آنها را در شکل ۲-۱۶ ملاحظه می‌کنید. باند C اولین باندی بود که به کاربردهای تجاری اختصاص یافت. این باند دو محدوده دارد: محدودهٔ پائینی که برای ارسال از ماهواره (downlink) استفاده می‌شود، و محدودهٔ بالایی که به ارسال به ماهواره (uplink) تخصیص یافته است. این باندها بسیار شلوغ هستند، چون در مخابرات مایکروویو زمینی هم از آنها استفاده می‌شود. باندهای L و S در سال ۲۰۰۰ طبق توافقهای بین‌المللی اضافه شدند، اما این باندها نیز بسیار باریک و شلوغ هستند.

باند بعدی که در اختیار مخابرات تجاری قرار دارد، باند Ku (K under) است. این باند هنوز پُر نشده، و در فرکانسهای آن می‌توان ماهواره‌ها را تا فاصلهٔ ۱ درجهٔ مداری به هم نزدیک کرد، ولی یک مشکل عمده دارد: باران. آب جاذب خوبی برای انرژی امواج در این طول موج است. خوشبختانه، طوفانهای بزرگ معمولاً به یک منطقهٔ کوچک محدود هستند، و اگر بجای یک ایستگاه زمینی از چندین ایستگاه با فواصل زیاد استفاده کنیم، این مشکل مرتفع خواهد شد (که البته این راه‌حل هزینهٔ بسیار بالایی دارد). باند Ka (K above) نیز به مصارف تجاری تخصیص یافته، ولی تجهیزات آن هنوز بسیار گران است. علاوه بر این باندهای تجاری، دهها باند دولتی و نظامی نیز وجود دارد.

یک ماهوارهٔ امروزی در حدود ۴۰ ترانسپاندر دارد، که پهنای باند هر کدام از آنها 80-MHz است. معمولاً هر ترانسپاندر بعنوان یک شیپور خمیده عمل می‌کند، ولی در ماهواره‌های جدیدتر (که امکانات پردازشی بیشتری دارند) می‌توان بگونه‌ای دیگر نیز عمل کرد. در ماهواره‌های اولیه، تقسیم ترانسپاندرها به کانالهای مختلف استاتیک بود: پهنای باند بصورت ساده به چند باند فرکانسی ثابت تقسیم می‌شد. امروزه، پرتو ترانسپاندر به بُرشهای زمانی

باند	دریافت از ماهواره	ارسال به ماهواره	پهنای باند	مشکلات
L	1.5 GHz	1.6 GHz	15 MHz	پهنای باند کم به شلوغ
S	1.9 GHz	2.2 GHz	70 MHz	پهنای باند کم به شلوغ
C	4.0 GHz	6.0 GHz	500 MHz	تداخل زمینی
Ku	11 GHz	14 GHz	500 MHz	باران
Ka	20 GHz	30 GHz	3500 MHz	باران یا قیمت بالای تجهیزات

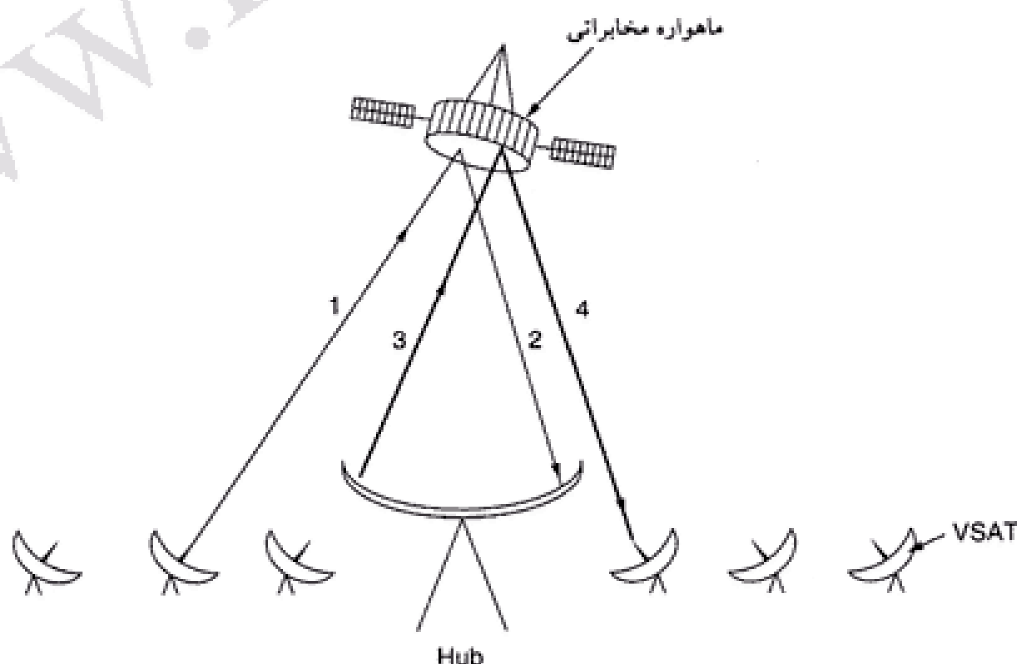
شکل ۲-۱۶. باندهای ماهواره‌ای.

تقسیم می‌شود، و هر کاربر می‌تواند بنوبت از آن استفاده کند. در قسمتهای آیندهٔ همین فصل این دو تکنیک (**مالتی‌پلکس تقسیم فرکانسی و مالتی‌پلکس تقسیم زمانی**) را به تفصیل بررسی خواهیم کرد.

اولین ماهواره‌های زمین‌ثابت یک پرتو فضایی واحد داشتند، که تقریباً $\frac{1}{3}$ سطح زمین را می‌پوشاند (به این ناحیه **جای پای ماهواره** - footprint - گفته می‌شود). اما با افت شدید قیمت، اندازه، و توان مصرفی وسایل میکروالکترونیک، استراتژیهای هوشمندانه‌تری امکان‌پذیر گردید. هر ماهواره به چندین آنتن و ترانسپاندر مجهز شد، و پرتو ارسالی از ماهواره چنان باریک شد که فقط منطقهٔ کوچکی را در بر می‌گرفت، و بدین‌ترتیب امکان مخابرهٔ همزمان چندین سیگنال بوجود آمد. این **پرتوهای نقطه‌ای** (spot beam) معمولاً بیضی شکل هستند، و می‌توان آنها را بقدری باریک کرد که فقط ناحیه‌ای به قطر چند صد کیلومتر را پوشش دهند. برای مثال، یک ماهوارهٔ مخابراتی که فقط برای ایالات متحدهٔ آمریکا به فضا پرتاب شده، را می‌توان طوری تنظیم کرد که یک پرتو برای ۴۸ ایالت خاک اصلی آمریکا، یک پرتو برای آلاسکا، و یکی برای هاوایی داشته باشد.

یکی از پیشرفتهای جدید در دنیای مخابرات ماهواره‌ای، ایستگاههای ارزان‌قیمتی هستند که VSAT (**ترمینال با باریکهٔ بسیار کوچک** - Very Small Aperture Terminal) نامیده می‌شوند (Abramson, 2000) . این ترمینالهای بسیار کوچک آنتنهایی بقطر ۱ متر یا کمتر دارند (برای مقایسه، قطر آنتنهای استاندارد GEO ۱۰ متر است)، و قدرت تشعشعی آنها در حدود ۱ وات است. سرعت ارسال این سیستمها در حدود 19.2-kbps است، ولی می‌توانند تا 512-kbps یا بیشتر دریافت کنند. تلویزیونهای ارسال مستقیم ماهواره‌ای (DB-SAT) معمولاً از این تکنولوژی استفاده می‌کنند.

در بسیاری از سیستمهای VSAT ، ایستگاههای زمینی بدلیل توان پائین نمی‌توانند مستقیماً با هم ارتباط برقرار کنند (البته از طریق ماهواره). در این موارد، از یک ایستگاه زمینی خاص، با آنتنی بزرگ و قوی بعنوان **هاب** (hub) ، برای رله کردن سیگنالها از یک VSAT به VSAT دیگر استفاده می‌کنند (شکل ۲-۱۷). در این حالت، یکی از طرفین باید آنتنی بزرگ با یک تقویت‌کنندهٔ قوی داشته باشد. زمان تأخیر سیگنال در این روش بیشتر از ارتباط



شکل ۲-۱۷. چند سیستم VSAT بهمراه یک هاب.

مستقیم است، ولی کاربران معمولاً هزینهٔ کمتر را به کمی تأخیر ترجیح می‌دهند.
سیستمهای VSAT بیشترین کاربرد را در مناطق روستایی دارد. متأسفانه، هنوز نیمی از جمعیت دنیا به تلفن دسترسی ندارند، و کشیدن خطوط تلفن به هزاران دهکدهٔ کوچک و دورافتاده از توان اغلب کشورهای جهان سوم خارج است – ولی نصب یک آنتن VSAT یک متری (که با سلولهای خورشیدی کار کند) کار چندان مشکلی نیست. VSAT تکنولوژیی است که می‌تواند همهٔ مردم دنیا را به هم وصل کند.
مخابرات ماهواره‌ای ویژگیهایی دارد که بشدت با ارتباطات نقطه–به–نقطهٔ زمینی متفاوتند. اولین ویژگی اینست که، با آنکه سیگنالهای ماهواره‌ای با سرعت نور (نزدیک 300,000 km/sec) حرکت می‌کنند، (بعلت فاصلهٔ زیاد ماهواره تا سطح زمین) ارتباط با ماهواره‌های GEO دارای زمان تأخیر سیگنال قابل‌ملاحظه‌ای است. بسته به فاصلهٔ کاربر از ایستگاه زمینی و زاویهٔ ماهواره نسبت به افق در آن محل، زمان ارتباط نقطه–به–نقطه بین ۲۵۰ تا ۳۰۰میلی‌ثانیه متغیر است. زمان متوسط این تأخیر ۲۷۰میلی‌ثانیه است (که برای سیستمهای VSAT دارای هاب به دو برابر، یعنی ۵۴۰میلی‌ثانیه، می‌رسد).
برای مقایسه بد نیست بدانید که، زمان تأخیر انتشار در لینکهای میکروویو زمینی حدود 3 μsec/km ، و برای کابلهای کواکسیال و فیبر نوری تقریباً 5 μsec/km است (امواج الکترومغناطیس در هوا سریعتر از مواد جامد حرکت می‌کنند).
ویژگی مهم دیگر ماهواره‌ها اینست که آنها ذاتاً رسانه‌های پخشی هستند؛ فرستادن پیام برای یک نفر هیچ تفاوتی با هزاران نفر ندارد. در برخی از کاربردها (مانند تبلیغات اینترنتی) این ویژگی بسیار مفید است. با اینکه بوسیلهٔ ارتباطات نقطه–به–نقطه هم می‌توان چنین وضعیتی را شبیه سازی کرد، ولی این کار با استفاده از ماهواره بسیار ارزانتر تمام می‌شود. از سوی دیگر، از دیدگاه امنیت و حفظ حریم خصوصی افراد، ماهواره یک فاجعهٔ تمام عیار است: هر کسی می‌تواند پیامهای خصوصی دیگران را بشنود. اینجاست که اهمیت **رمزنگاری** (encryption) روشن می‌شود.
در مخابرات ماهواره‌ای، هزینهٔ انتقال پیام به فاصلهٔ فرستنده و گیرنده بستگی ندارد: تماس با آن سوی اقیانوسها از نظر هزینه هیچ فرقی با تماس با خانهٔ روبرویی ندارد. مخابرات ماهواره‌ای از نظر نرخ خطا بسیار عالیست، و زمان به بهره‌برداری رسیدن آن نیز بسیار کوتاه است (نکته‌ای که در مخابرات نظامی بسیار اهمیت دارد).

۲-٤-۲ ماهواره‌های مدار متوسط

در مداری بسیار پائینتر از مدار زمین‌ثابت، و بین دو کمربند وان‌الن، **ماهواره‌های مدار متوسط** که به MEO (Medium-Earth Orbit) معروفند، قرار می‌گیرند. این ماهواره‌ها بطور متوسط هر ۶ ساعت یکبار دور زمین می‌گردند، و بهمین دلیل ایستگاه زمینی باید آنها را تعقیب کند. بعلت ارتفاع پائین، ماهواره‌های MEO جای پای کوچکتری نسبت به ماهواره‌های GEO دارند، ولی توان تشعشعی لازم برای ارسال به آنها نیز بسیار کمتر است. در حال حاضر از این ماهواره‌ها برای مقاصد مخابراتی استفاده نمی‌شود، بنابراین ما هم دربارهٔ آنها بیش از این صحبت نخواهیم کرد. ماهواره‌های ۲۴گانهٔ GPS (سیستم مکان‌یابی جهانی – Global Positioning System) که در ارتفاع 18,000 km پرواز می‌کنند، در این دسته قرار می‌گیرند.

۳-٤-۲ ماهواره‌های مدار پائین

اگر باز هم پائینتر بیائیم، به **ماهواره‌های مدار پائین** LEO (Low-Earth Orbit) می‌رسیم. بدلیل سرعت زیاد گردش مداری این ماهواره‌ها، برای ایجاد سیستمی با پوشش جهانی به تعداد زیادی از آنها نیاز داریم. از طرف دیگر، بدلیل ارتفاع کم ماهواره‌های LEO ، برای ارتباط با آنها به توان کمی نیاز است، و زمان رفت و برگشت

سیگنال نیز بسیار کم (در حدود چند میلی‌ثانیه) خواهد بود. در این قسمت سه سیستم ماهواره‌ای LEO را - که دو تای آنها به سرویس صدا و یکی به سرویسهای اینترنت اختصاص دارند - مورد بررسی قرار خواهیم داد.

ایریدیوم

همانطور که گفتیم بعلت سرعت زیاد گردش مداری ماهواره‌های LEO - و عبور سریع از مقابل آنتنهای زمینی - تا همین اواخر از این ماهواره‌ها برای مقاصد مخابراتی استفاده نمی‌شد. این وضعیت در سال ۱۹۹۰ تغییر کرد: در این سال **موتورولا** درخواستی برای دریافت مجوز پرتاب ۷۷ ماهوارهٔ مخابراتی LEO به FCC ارائه کرد - این پروژه **ایریدیوم** (Iridium) نام داشت (ایریدیوم عنصر ۷۷ام جدول تناوبی است). بعدها این پروژه تغییر کرد، و قرار شد از فقط ۶۶ ماهواره استفاده شود - بالطبع نام پروژه را هم باید به **دیسپروسیوم** (عنصر ۶۶ام جدول تناوبی) تغییر می‌دادند، اما نام دیسپروسیوم بیشتر شبیه یک بیماری خطرناک است تا یک سیستم ماهواره‌ای! ایدهٔ اصلی در این سیستم آن است که به محض خارج شدن یک ماهواره از دید آنتن زمینی، ماهوارهٔ دیگری بلافاصله جای آنرا می‌گیرد. این پیشنهاد به یک هیجان عمومی در میان شرکتهای مخابراتی دامن زد، و همه خواهان آن بودند که زنجیره‌ای از ماهواره‌های LEO به فضا پرتاب کنند.

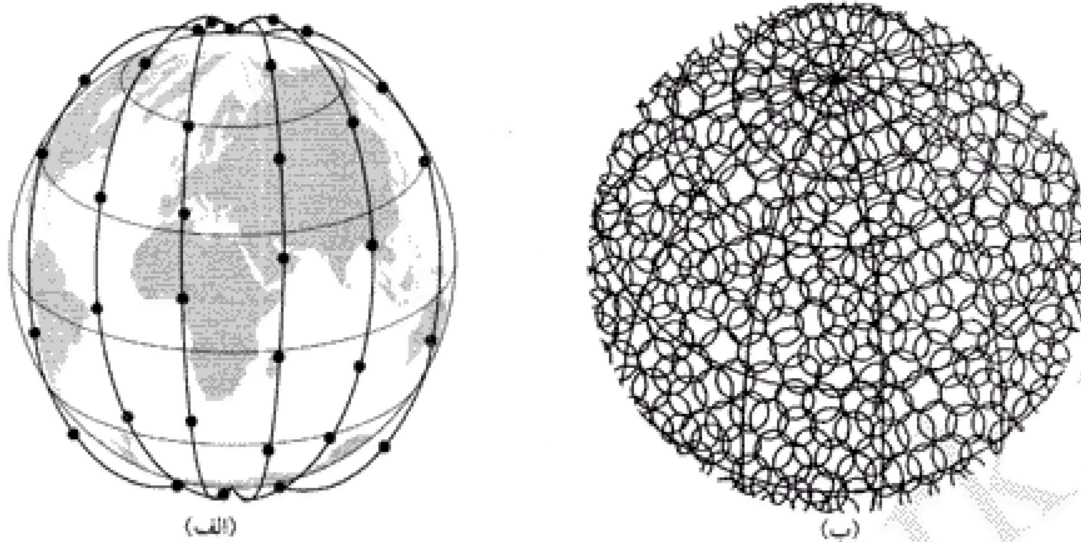
بعد از هفت سال اشتراک مساعی فنی و مالی بین چند شرکت بزرگ، بالاخره ماهواره‌های ایریدیوم در سال ۱۹۹۷ به فضا پرتاب شدند، و از نوامبر ۱۹۹۸ سرویسهای مخابراتی آنها شروع بکار کرد. اما متأسفانه بدلیل گسترش شبکهٔ تلفنهای همراه، تقاضای ناچیزی برای این سرویسها در بازار وجود داشت، و متعاقب آن در آگوست ۱۹۹۹ پروژهٔ ایریدیوم (طی یکی از فاجعه‌بارترین ورشکستگی‌های تاریخ) متوقف شد. ماهواره‌ها و سایر تجهیزات این پروژه (که ۵ میلیارد دلار ارزش داشت) به قیمت ۲۵ میلیون دلار به یک سرمایه‌گذار فروخته شد؛ در واقع، پروژهٔ ایریدیوم را باید بزرگترین بازار اسقاطی فضایی بشمار آورد. سرویسهای ایریدیوم از مارس ۲۰۰۱ دوباره راه‌اندازی شد.

کار اصلی ایریدیوم ارائهٔ سرویسهای مخابراتی از طریق تجهیزاتی که مستقیماً با ماهواره ارتباط برقرار می‌کنند، بود (و هست). ایریدیوم سرویسهای صدا، داده، فکس، پیجر، و هدایت و ناوبری را در تمام نقاط زمین (خشکی، دریا و هوا) در اختیار کاربران خود می‌گذارد. کاربران این سرویسها را عمدتاً دریانوردان، هوانوردان، کارکنان سکوهای اکتشاف نفت، و مسافران و جهانگردانی که به مناطق فاقد زیرساختهای مخابراتی (مانند کوه، صحرا، جنگل، و برخی از کشورهای جهان سوم) سفر می‌کنند، تشکیل می‌دهند.

ماهواره‌های ایریدیوم در مدار قطبی و در ارتفاع ۷۵۰کیلومتری زمین پرواز می‌کنند. این ماهواره‌ها بصورت کمربندهایی که از قطبهای شمال-جنوب زمین می‌گذرند، آرایش یافته‌اند، و با یکدیگر ۳۲ درجه عرض جغرافیایی فاصله دارند - شکل ۲-۱۸ (الف) را ببینید. برای پوشش دادن تمام سطح زمین شش تا از این حلقه‌ها کفایت می‌کند. آنهایی که کمی با شیمی آشنایند، می‌توانند این ماهواره‌ها را الکترونهای یک اتم غول‌آسای دیسپروسیوم فرض کنند که بدور هستهٔ اتم (کرهٔ زمین) در گردشند.

هر ماهواره حداکثر ۴۸ سلول (پرتو نقطه‌ای) دارد، که بدین‌ترتیب تعداد کل سلولها به ۱۶۲۸ می‌رسد، و می‌توانند کل سطح زمین را پوشش دهند - شکل ۲-۱۸ (ب) را ببینید. هر ماهوارهٔ ایریدیوم ۳۸۴۰ کانال - و کل سیستم ۲۵۳,۴۴۰ کانال - ظرفیت دارد، و می‌توان از آنها برای سرویسهای مختلف استفاده کرد.

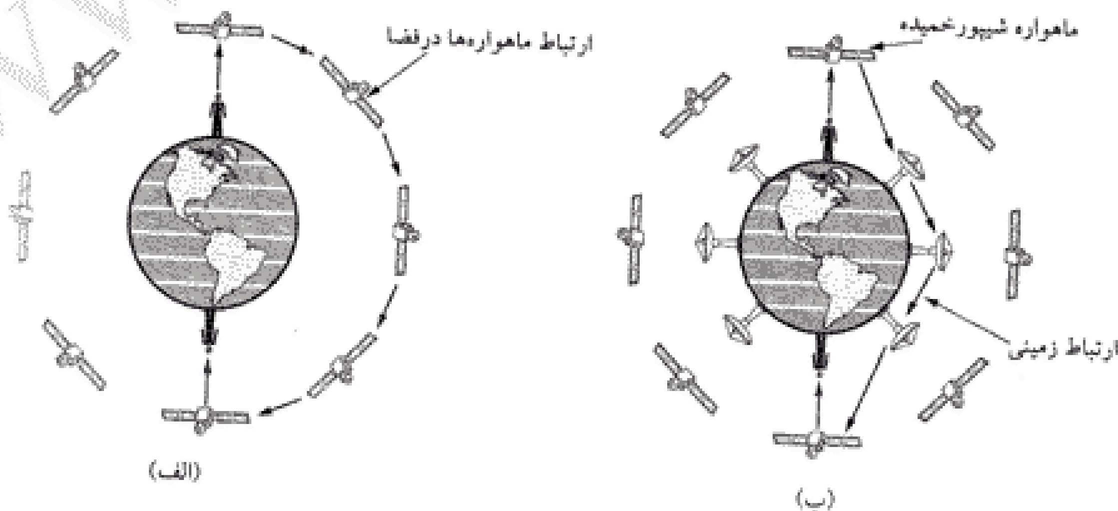
یکی از نکات جالب در مورد ایریدیوم اینست که ارتباط کاربرانی که از یکدیگر فاصلهٔ زیادی دارند، در فضا صورت می‌گیرد (بعبارت دیگر، رله کردن اطلاعات بین ماهواره‌ها در فضا انجام می‌شود)؛ شکل ۲-۱۹(الف) را ببینید. همانطور که در این شکل می‌بینید، وقتی کاربری در قطب شمال بخواهد با قطب جنوب تماس بگیرد، سیگنال را به ماهواره‌ای که بالای سرش قرار دارد، می‌فرستد. سپس این سیگنال از یک ماهواره به ماهوارهٔ دیگر رله می‌شود، تا به قطب جنوب برسد.



شکل ۲-۱۸. (الف) ماهواره‌های ایریدیوم شش حلقه بدور زمین می‌سازند. (ب) تعداد سلولهای متحرک ایریدیوم به ۱۶۲۸ می‌رسد.

گلوبال‌استار

یکی از طرحهای رقیب ایریدیوم، پروژهٔ **گلوبال‌استار** (Globalstar) است. این پروژه از ۴۸ ماهوارهٔ LEO تشکیل شده، ولی روش سوئیچینگ آن با ایریدیوم فرق دارد، و در آن از سوئیچینگ زمینی استفاده می‌شود - شکل ۲-۱۹ (ب) را ببینید. در اینجا سیگنالها از ماهوارهٔ مبدأ به یک ایستگاه زمینی بزرگ واقع در سانتا ورک‌شاپ هدایت شده، و از آنجا به نزدیکترین ایستگاه زمینی نزدیک مقصد فرستاده می‌شوند، تا بالاخره (بعد از ارسال به ماهواره) به گیرنده برسند. مزیت این روش آنست که (بر خلاف ایریدیوم) قسمت پیچیدهٔ کار روی زمین انجام می‌شود، و مدیریت آن ساده‌تر است. همچنین، با آنتنهای بزرگ زمینی (که سیگنالهای قویتری می‌فرستند، و می‌توانند سیگنالهای ضعیفتری دریافت کنند) از تجهیزات انفرادی ساده‌تری می‌توان استفاده کرد (توان تشعشعی تلفنها در حد چند میلی‌وات است، و حتی بعد از تقویت در ماهواره باز هم چندان قوی نیست).



شکل ۲-۱۹. (الف) رلهٔ سیگنال در فضا. (ب) رلهٔ سیگنال روی زمین.

تله‌دزیک

کاربران اصلی ایریدیوم افرادی هستند که به جاهای پرت و دورافتاده می‌روند. نمونهٔ دیگر سیستمهای ماهواره‌ای **تله‌دزیک** (Teledesic) است، که کاربران اینترنت پُرسرعت را (در تمام دنیا) هدف گرفته است. ایدهٔ این سیستم به کریگ مک‌کاو (پیشگام صنعت تلفنهای همراه) و بیل گیتس (بنیانگذار میکروسافت) تعلق دارد، که از سرعت لاک‌پشتی دسترسی اینترنت ناراضی بودند. هدف سیستم تله‌دزیک فراهم آوردن دسترسی به اینترنت با سرعتهای بالا (تا 100-Mbps ارسال، و تا 720-Mbps دریافت) برای میلیونها کاربر همزمان است. این سیستم از آنتنهای کوچک و ثابت (شبیه VSAT) استفاده می‌کند، و هیچ نیازی به ارتباطات تلفنی معمولی ندارد.

در طرح اولیهٔ سیستم تله‌دزیک قرار بود از ۲۸۸ ماهوارهٔ پرتو باریک، که در دوازده گروه در ارتفاع ۱۳۵۰ کیلومتری (درست زیر کمربند وان‌آلن تحتانی) پرواز می‌کنند، استفاده شود. ولی این طرح بعدها به ۳۰ ماهواره با جای پای بزرگتر تغییر کرد. ماهواره‌ها در باند نسبتاً خلوت Ka و بصورت سوئیچینگ بسته‌ای کار خواهند کرد، و می‌توانند بسته‌ها را بین خود رد و بدل کنند. وقتی یک کاربر به پهنای باند نیاز پیدا می‌کند، درخواست خود را بصورت یک بسته می‌فرستد، و ۵۰ میلی‌ثانیه بعد پهنای باند موردنیاز بصورت خودکار به وی اختصاص داده می‌شود. اگر همه چیز طبق نقشه پیش برود، این سیستم در سال ۲۰۰۵ عملیاتی خواهد شد.

۲-٤-٤ ماهواره یا فیبر؟

مقایسه‌ای بین مخابرات ماهواره‌ای و زمینی آموزنده خواهد بود. تا همین ۲۰ سال پیش هیچکس شک نداشت که آیندهٔ مخابرات متعلق به ماهواره‌هاست، چون سیستمهای تلفن در ۱۰۰ سال گذشته تغییر عمده‌ای نکرده بودند، و بنظر نمی‌رسد در ۱۰۰ سال بعدی هم هیچ اتفاق خاصی بیفتد. شرکتهای تلفن سرویسهای صوتی خوب (با قیمت مناسب) ارائه می‌کردند، و سرمایه‌گذاری آنها سودی تضمین شده داشت. برای آنهایی هم که سرویس داده می‌خواستند، مودمهای 1200-bps موجود بود - و این تمام بضاعت شرکتهای تلفن بود.

اما در سال ۱۹۸۴ آتش رقابت ایالات متحدهٔ آمریکا و اروپا را در بر گرفت، و اوضاع تغییر کرد. شرکتهای تلفن در سرویسهای راه‌دور فیبرهای نوری را جایگزین کابلهای مسی کردند، و سرویسهای با پهنای باند زیاد مانند ADSL (**خط دیجیتال نامتقارن**) در اختیار کاربران خود قرار دادند. تغییر اساسی دیگر کاهش نرخهای مخابرات راه‌دور بود، که تا آن زمان بطور مصنوعی و به نفع کاربران محلی بالا نگه داشته شده بود. بنظر می‌رسید فیبر نوری برندهٔ جنگ باشد، ولی ماهواره هم مزایایی دارد که فیبر نمی‌تواند (و احتمالاً نخواهد توانست) با آنها رقابت کند.

اول اینکه، پهنای باند فیبر (که شاید یک رشتهٔ آن از تمام ماهواره‌های پرتاب شده بیشتر باشد) در اختیار اغلب کاربران نیست. فیبر نوری بیشتر در مخابرات راه‌دور و برای متصل کردن شبکه‌های تلفن بکار می‌رود، تا رساندن پهنای باند بالا به کاربران منفرد. در حالیکه ماهواره‌ها (به کمک یک آنتن ساده در بالای پشت بام) می‌توانند مستقیماً و بدون هیچ واسطه‌ای پهنای باند زیاد را در اختیار تک تک کاربران قرار دهند. سیستم تله‌دزیک بر اساس همین ایده شکل گرفته است.

مخابرات سیار دیگر قلمرو دست‌نیافتنی ماهواره‌هاست. امروزه بسیاری از افراد مایلند در حال قدم‌زدن، اتومبیل‌سواری، و حتی قایقرانی و پرواز به سرویسهای مخابراتی دسترسی داشته باشند. اینجا دیگر فیبر نوری بکلی بی‌استفاده است، و فقط مخابرات ماهواره‌ای می‌تواند راه چاره باشد. البته برای آنهایی که شعاع حرکتشان محدود است، می‌توان از ترکیب فیبر و بیسیمهای رادیویی استفاده کرد، ولی در هوا و دریا این روش هم دیگر کارایی ندارد.

مزیت دیگر ماهواره در مخابرات **پخشی** (broadcasting) است. وقتی بخواهید یک پیام را در آن واحد به هزاران نفر برسانید (مانند ارسال نرخ کالا و ارز، یا قیمت سهام و اوراق قرضه)، هیچ چیز جای ماهواره را (از نظر

سهولت و هزینه) نمی‌گیرد.

پراکندگی جمعیت و سرزمین از دیگر عواملیست که بکارگیری مخابرات ماهواره‌ای را مقرون بصرفه می‌کند. برای مثال، کشور اندونزی برای هدایت ترافیک تلفن داخلی خود نیز از ماهواره استفاده می‌کند، چون پرتاب یک ماهواره بسیار ساده‌تر است، تا کشیدن کابلهای زیردریایی به ۱۳۶۷۷ جزیره‌ای که این کشور را تشکیل می‌دهند. در جاهایی که تملک زمین برای کشیدن کابل زمینی دشوار (و یا پُرهزینه) است، نیز ماهواره مزیت نسبی دارد.

سرعت نصب و راه‌اندازی سیستم نیز یکی از جاهائیست که ماهواره بر کابل زمینی پیروز می‌شود. وقتی جنگی در می‌گیرد، و نیروهای نظامی به تماس با نقاط جدید نیاز پیدا می‌کنند، پرتاب یک ماهواره معمولاً سریعترین راه‌حل ممکن است.

بطور خلاصه، بنظر می‌رسد که جریان اصلی مخابرات در آینده بر فیبرهای نوری و تلفنهای همراه متکیست، ولی در برخی از کاربردهای خاص ماهواره‌ها برتری دارند. اما همهٔ اینها تابع یک چیز هستند: اقتصاد. با اینکه فیبر نوری پهنای باند بیشتری ارائه می‌کند، اما حرف آخر را قیمت می‌زند. اگر تکنولوژی جدیدی اختراع شود که هزینهٔ پرتاب ماهواره‌ها را بشدت کاهش دهد (مانند شاتلهای فضایی که بتوانند هر بار دهها ماهواره را به مدار ببرند)، یا کاربرد غیرمنتظره‌ای برای ماهواره‌های مدار پائین پیدا شود، معلوم نیست که فیبر نوری در تمام زمینه‌ها برنده باشد.

۲ـ۵ شبکهٔ تلفن عمومی

اگر بخواهیم دو کامپیوتر را که نزدیک به هم و یا در یک ساختمان هستند، به یکدیگر متصل کنیم، کار بسیار ساده است و فقط کافیست یک رشته کابل بین آنها بکشیم - این همان شبکهٔ محلی یا LAN است. اما اگر فاصلهٔ کامپیوترها زیاد باشد، یا کابل بایستی از املاک خصوصی یا شهری عبور داده شود، هزینهٔ انجام کابل‌کشی معمولاً به مانعی بزرگ تبدیل می‌شود (البته اگر این کار غیرقانونی نباشد، که در اغلب کشورها هست). در نتیجه، طراحان شبکه به تأسیسات مخابراتی موجود روی می‌آورند.

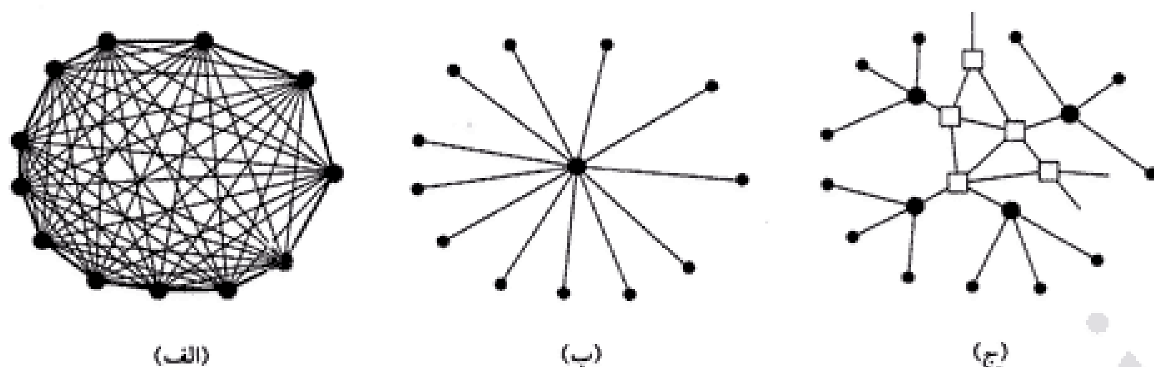
این تأسیسات بویژه **شبکهٔ تلفن عمومی** - PSTN (Public Switched Telephone Network) - سالها قبل و با هدفی دیگر طراحی شده‌اند: انتقال صدای انسان بگونه‌ای کمابیش قابل تشخیص. کارایی این تجهیزات برای ارتباط کامپیوتر-کامپیوتر در بهترین حالت اغلب حاشیه‌ای است، اما با به بازار آمدن فیبر نوری و تکنولوژیهای دیجیتال این وضعیت بسرعت در حال تغییر است. در هر حال، سیستم تلفن عمومی چنان با شبکه‌های (گستردهٔ) کامپیوتری عجین است، که جا دارد وقت بیشتری به بررسی آن اختصاص دهیم.

برای بهتر شکافتن صورت مسئله، اجازه دهید مقایسه‌ای کلی (ولی روشن‌کننده) بین ارتباط دو کامپیوتر از طریق کابل مستقیم شبکه و از طریق خطوط تلفن داشته باشیم. کابل مستقیم شبکه می‌تواند داده‌ها را با سرعتی معادل 10^9 bps (یا حتی بیشتر) منتقل کند، در حالیکه حداکثر سرعت یک خط تلفن 56-kbps بیشتر نیست - تفاوتی در حد ۲۰,۰۰۰ برابر. تفاوت این دو مانند تفاوت سرعت یک مرغابی که سلانه سلانه در علفزار راه می‌رود، با موشکی است که به طرف ماه پرواز می‌کند. اگر به جای خط تلفن از اتصال ADSL استفاده کنیم، تفاوت سرعت به ۱۰۰۰ تا ۲۰۰۰ برابر خواهد رسید.

ناگفته پیداست که چنین سطح از تفاوتی برای طراحان سیستمهای کامپیوتری مشکل‌ساز است، و آنان تمام تلاش خود را برای بهینه کردن آن متمرکز کنند. در قسمتهای آینده سیستم تلفن و طرز کار آن را تشریح خواهیم کرد؛ برای اطلاعات بیشتر در این زمینه به (Bellamy, 2000) مراجعه کنید.

۲ـ۵ـ۱ ساختار سیستم تلفن

بلافاصله بعد از آن که آلکساندر گراهام بل در سال ۱۸۷۶ (و درست چند ساعت زودتر از رقیبش، الیشا گری)



شکل ۲-۲۰. (الف) شبکه‌ای با اتصالات داخلی کامل. (ب) سوئیچ مرکزی. (ج) سلسله مراتب دو سطحی.

اختراع خود را به ثبت رساند، سیل تقاضا برای آن سرازیر شد. تلفنهای اولیه صورت جفتی کار می‌کردند، و مشتریان مجبور بودند بین خودشان یک رشته سیم بکشند (مسیر برگشت الکترونها از زمین بود). اگر کسی می‌خواست با n نفر تماس تلفنی داشته باشد، مجبور بود به این n نقطه سیم بکشد. در کمتر از یک سال شهرها تبدیل شدند به جنگلی از سیمهای تو در تو که از خانه‌ای به خانهٔ دیگر (از فراز ساختمانها و درختها) کشیده شده بود. خیلی زود معلوم شد که مدل اتصال هر تلفن به تمام تلفنهای دیگر مدلی عملی نیست - شکل ۲-۲۰ (الف).

خوشبختانه بل خیلی زود متوجه این مشکل شد، و با تأسیس شرکتی بنام **شرکت تلفن بل**، اولین مرکز سوئیچینگ را در سال ۱۸۷۸ (در نیوهیون، کانتیکات) راه‌اندازی کرد. این شرکت یک رشته کابل به خانه یا دفتر هر مشتری می‌کشید. برای برقراری تماس، مشتری دستهٔ تلفن را می‌چرخاند تا زنگی در مرکز تلفن بصدا در آید و توجه اپراتور جلب شود؛ سپس این اپراتور ارتباط وی را توسط یک رشته کابل با مقصد موردنظر برقرار می‌کرد. این مدل را در شکل ۲-۲۰ (ب) ملاحظه می‌کنید.

بزودی در هر شهر و دهکده‌ای مراکز سوئیچینگ بل دایر شد، و بل مجبور شد برای برقرای تماسهای راه دور این مراکز را نیز به یکدیگر متصل کند. اما در اینجا هم همان مشکل قبلی خود را نشان داد: ارتباط مستقیم هر مرکز سوئیچنگ با تمام مراکز دیگر غیرممکن بود، بنابراین مراکز سوئیچینگ سطح-دوم اختراع شد. پس از مدتی کوتاه تعداد مراکز سطح-دوم نیز بشدت افزایش یافت - شکل ۲-۲۰ (ج) را ببینید. بعدها این سلسله مراتب تا پنج سطح بالا رفت.

تا سال ۱۸۹۰ این سیستم تلفن سه بخش عمده داشت: مراکز سوئیچینگ، سیمهایی که بین مراکز سوئیچینگ و مشترکان کشیده می‌شد (این سیمها دیگر سیمهای لخت با برگشت زمین نبود، بلکه تبدیل به سیمهای زوج تابیدهٔ عایق‌دار شده بود)، و اتصالات راه دور بین مراکز سوئیچینگ. با آن که پیشرفتهایی در هر یک از این سه بخش صورت گرفته، اما مدل اولیهٔ بل در ۱۰۰ سال گذشته تقریباً بدون تغییر باقی مانده است. برای دیدن تاریخچه‌ای مختصر از سیستم تلفن بل، به (Hawley, 1991) نگاه کنید.

تا قبل از دو پاره شدن AT&T در سال ۱۹۸۴، سیستم تلفن سیستمی با سلسله مراتب چندسطحی (و با پراکندگی زیاد) بود. در بحث زیر این ساختار تا حد زیادی ساده شده، ولی عصارهٔ اصلی آن همچنان حفظ شده است. هر تلفن با دو رشته سیم مسی به نزدیکترین **ایستگاه پایانی** (local central office یا end office) وصل می‌شود. فاصلهٔ این دو معمولاً بین ۱ تا ۱۰ کیلومتر است (و در شهرها کمتر از مناطق روستائیست). فقط در ایالات متحدهٔ آمریکا نزدیک به ۲۲,۰۰۰ ایستگاه پایانی وجود دارد. به اتصال دو-سیمهٔ بین ایستگاه پایانی و مشترک تلفن

اصطلاحاً **مدار پایانی** (local loop) گفته می‌شود. اگر مدارهای پایانی موجود در سراسر دنیا را بدنبال هم ردیف کنند، می‌تواند هزار بار فاصلهٔ زمین تا ماه را بپیماید.

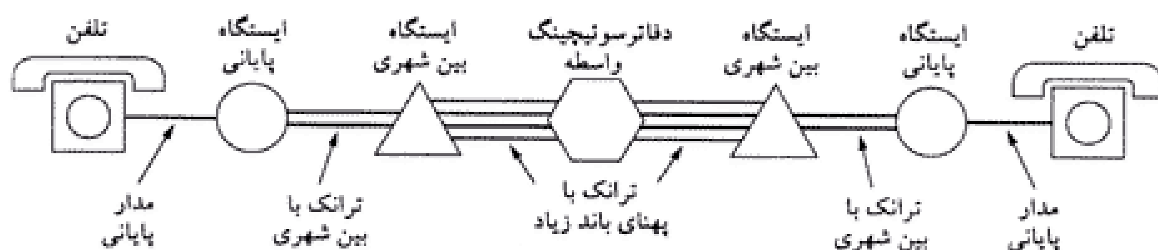
طبق یکی از تخمین‌ها، ۸۰ درصد ارزش سرمایهٔ AT&T مس موجود در مدارهای پایانی آن است - بدین‌ترتیب، می‌توان AT&T را بزرگترین معدن مس دنیا بحساب آورد. خوشبختانه بورس کالا از این موضوع اطلاع چندانی ندارد - اگر اینرا می‌دانستند، بلافاصله AT&T را می‌خریدند، تمام سرویسهای تلفن را قطع می‌کردند، مس‌ها را در می‌آوردند، و دوباره به صنایع مس می‌فروختند.

وقتی یک مشترک به مشترک دیگری که به همان ایستگاه پایانی وصل است تلفن می‌زند، دستگاههای سوئیچینگ بین این دو مدار پایانی یک ارتباط الکتریکی مستقیم برقرار می‌کنند- این ارتباط مستقیم در تمام طول تماس برقرار می‌ماند.

اما اگر مشترک موردنظر متعلق به ایستگاه پایانی دیگری باشد، روش کار فرق می‌کند. هر ایستگاه پایانی دارای چند خط ارتباطی با یک یا چند مرکز سوئیچینگ، که اصطلاحاً **ایستگاه بین‌شهری** (toll office) نامیده می‌شوند. است - اگر این ایستگاه‌ها در یک منطقه باشند، به آنها **ایستگاه شریک** (tandem office) نیز گفته می‌شود. این خطوط ارتباطی را **ترانک‌های مرتبط‌کنندهٔ بین‌شهری** (toll connecting trunks) می‌نامند. اگر ایستگاههای پایانی تلفن‌کننده و تلفن‌شونده به یک ترانک وصل باشند (که در صورت نزدیکی آنها بسیار محتمل است)، امکان دارد ارتباط در همان ایستگاه بین‌شهری برقرار شود. در شکل ۲-۲۰ (ج) یک شبکهٔ تلفن ساده را می‌بینید، که در آن تلفنها با نقاط سیاه کوچک، ایستگاههای پایانی با نقاط سیاه بزرگ، و ایستگاه‌های بین‌شهری با مربع نشان داده شده‌اند.

اگر تلفن‌کننده و تلفن‌شونده دارای ایستگاه بین‌شهری مشترک نباشند، مسیر ارتباطی بایستی در سطح بالاتری برقرار شود. ایستگاه‌های بین‌شهری از طریق ایستگاه‌های **اولیه** (primary office)، **ناحیه‌ای** (sectional office) و **منطقه‌ای** (regional office) به هم متصل می‌شوند. اتصال این ایستگاه‌ها از طریق **ترانک بین‌شهری** (intertoll trunk یا interoffice trunk) برقرار می‌شود. نوع مراکز سوئیچینگ و توپولوژی آنها (مثلاً، اینکه دو ایستگاه ناحیه‌ای می‌توانند مستقیماً به یکدیگر وصل شوند، یا باید این ارتباط از طریق یک ایستگاه منطقه‌ای باشد؟) از کشوری به کشور دیگر فرق می‌کند، و به تراکم تلفن در آن کشور بستگی دارد. در شکل ۲-۲۱ نحوهٔ هدایت یک تماس راه دور را ملاحظه می‌کنید.

در مخابرات راه دور از رسانه‌های مختلفی استفاده می‌شود. امروزه در مدارهای پایانی از کابل Cat 3 استفاده می‌شود، در حالیکه در سالهای اولیهٔ اختراع تلفن از سیمهای بدون پوششی که با فاصلهٔ ۲۵ سانتیمتر بر فراز تیرهای تلفن کشیده می‌شد، استفاده می‌کردند. بین مراکز سوئیچینگ معمولاً کابل کواکسیال، مایکروویو یا فیبر نوری بکار برده می‌شود.



شکل ۲-۲۱. مدار هدایت یک تماس راه دور.

در گذشته انتقال سیگنالهای تلفن بصورت آنالوگ بود: ابتدا صدا به ولتاژ الکتریکی تبدیل شده، و سپس همین سیگنال روی خط تلفن ارسال می‌شد. با اختراع فیبر نوری، الکترونیک دیجیتال و کامپیوتر، امروزه دیگر تمام‌ترانک‌ها و سوئیچها دیجیتال هستند، و تنها قسمتی که هنوز از تکنولوژی آنالوگ استفاده می‌کند، همان مدار پایانی است. مزیت انتقال دیجیتال در اینست که دیگر مشکل اعوجاج سیگنال در تقویت‌کننده‌های مختلف وجود ندارد، و فقط کافیست بتوانیم 0 را از 1 تشخیص دهیم. انتقال دیجیتال مطمئنتر، ارزانتر، و نگهداری آن ساده‌تر است.

بطور خلاصه، سیستم تلفن از سه قسمت عمده تشکیل شده است:

۱. مدارهای پایانی (زوجهای تابیدهٔ آنالوگ که به خانه‌ها و دفاتر کشیده می‌شوند)
۲. ترانک‌ها (فیبرهای نوری دیجیتال که مراکز سوئیچینگ را به یکدیگر متصل می‌کنند)
۳. مراکز سوئیچینگ (مراکزی که تماسهای تلفنی را از یک خط اصلی به خط دیگر هدایت می‌کنند)

اجازه دهید قبل از پرداختن به ادامهٔ این بحث، کمی هم دربارهٔ تلفن و سیاست صحبت کنیم. بعد از آن خواهیم دید که مدارهای پایانی و ترانک‌ها چگونه عمل می‌کنند، و سوئیچینگ چگونه انجام می‌شود.

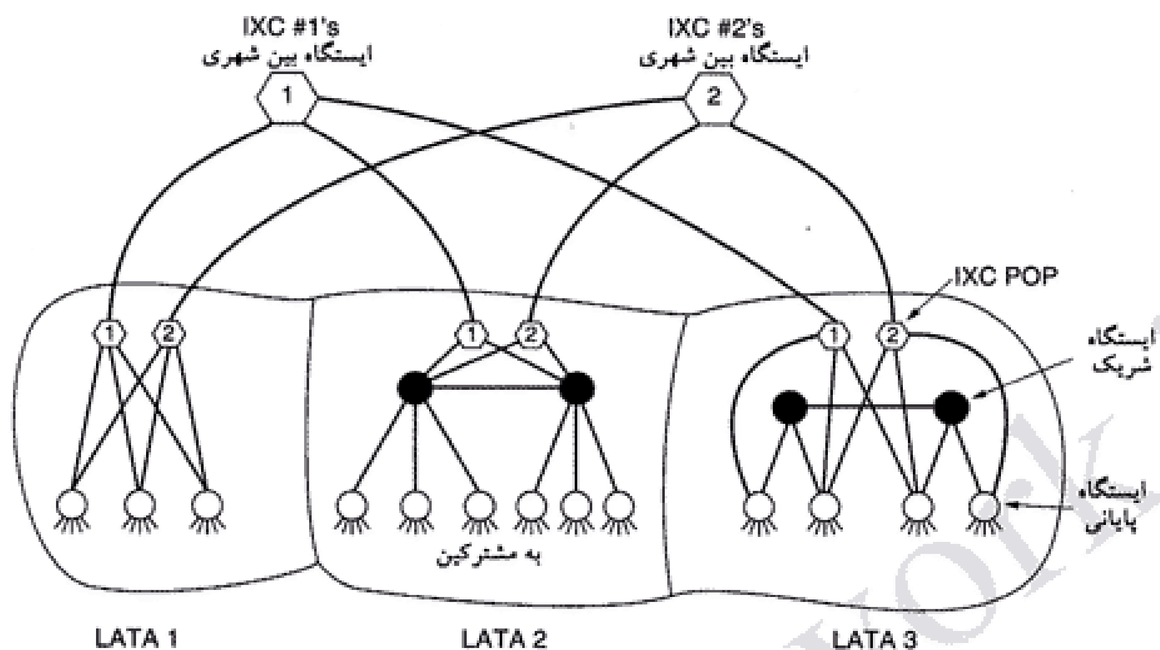
۲-۵-۲ تلفن و سیاست

تا قبل از سال ۱۹۸۴ برای چندین دهه انحصار تلفن شهری و راه دور ایالات متحدهٔ آمریکا در انحصار شرکت **بل سیستم** بود. در دههٔ ۱۹۷۰ دولت فدرال به این نتیجه رسید که این انحصار غیرقانونی است، و برای شکستن آن به دادگاه شکایت کرد. در اول ژانویه ۱۹۸۴ رأی دادگاه به نفع دولت فدرال صادر شد، و شرکت AT&T به شرکتهای AT&T Long Lines ، ۲۳ BOC (شرکتهای Bell Operating Company)، و چند شعبهٔ کوچک دیگر شکسته شد. شرکتهای BOC برای آن که بتوانند در بازار رقابت کنند، هفت گروه منطقه‌ای (RBOC) تشکیل دادند. این رأی (که به Modified Final Judgment - MFJ - معروف است) مخابرات راه دور ایالات متحده را یک شبه دچار دگرگونی اساسی کرد.

دستور MFJ منجر به تغییرات مهمی در سرویسهای مخابراتی شد: افزایش رقابت بین شرکتهای مخابراتی، بهبود سرویسها، و کاهش قیمت مخابرات راه دور. از طرف دیگر قیمت سرویسهای شهری بشدت بالا رفت، چون این شرکتها بایستی بنوعی کاهش درآمد خود را (که قبلاً از درآمد مخابرات راه دور تأمین می‌شد) جبران می‌کردند. بسیاری از کشورهای دیگر نیز در حال پیروی از این مدل هستند.

برای آن که مشخص شود چه کسی مجاز به انجام چه کاریست، ایالات متحده به ۱۶۴ LATA (**مناطق محلی دسترسی و انتقال** - Local Access and Transport Area) تقسیم شد. تقسیمبندی LATA ها تا حد زیادی (اما نه کاملاً) بر **کُدهای منطقه** (area code) منطبق است. هر LATA دارای یک LEC (**کاریر تبادل محلی** - Local Exchange Carrier) است که انحصار کامل سرویسهای تلفن در آن منطقه را در اختیار دارد. مهمترین این LEC ها همان BOC ها هستند، ولی در برخی از مناطق شرکتهای مستقل نیز (که تعداد آنها به ۱۵۰۰ می‌رسد) فعالیت دارند.

کلیهٔ ترافیک بین LATA ها توسط شرکتهایی بنام IXC (**کاریر تبادلی** - InterеXchange Carrier) انجام می‌شود. تا مدتی قبل AT&T Long Lines تنها IXC فعال در بازار بود، ولی اکنون شرکتهای بزرگی مانند WorldCom و Sprint نیز در این زمینه فعال هستند. یکی از دغدغه‌هایی که بعد از شکسته شدن AT&T وجود داشت آن بود که IXC ها در زمینهٔ کیفیت خط، تعرفه‌ها، و تعداد رقمهای پیش‌شماره یکسان باشند. روش این کار در شکل ۲-۲۲ نشان داده شده است؛ در این شکل سه LATA با تعدادی ایستگاه پایانی می‌بینید. LATA های ۲ و ۳ با ایستگاه‌های بین‌شهری نیز ارتباط دارند.



شکل ۲-۲۲. ارتباط بین LATAها و LECها و IXCها

هر IXC که بخواهد مجری تماسهای یک LATA باشد، یک ایستگاه سوئیچینگ بنام POP (**نقطهٔ تماس - Point Of Presence**) در آنجا تأسیس می‌کند. وظیفهٔ برقراری تماس IXC با ایستگاههای پایانی بر عهدهٔ LEC آن منطقه است (خواه بصورت مستقیم مانند LATAهای ۱ و ۳، یا بصورت غیرمستقیم مانند LATA ۲). تماس (خواه فنی یا مالی) باید برای تمام IXC ها یکسان باشد. بدین‌ترتیب، مشترکی که مثلاً در LATA ۱ است، می‌تواند تصمیم بگیرد توسط کدام IXC با مشترکی در LATA ۳ تماس بگیرد.

در MFJ تصریح شده است که IXC ها نباید وارد بازار تلفن محلی شوند، و LEC ها هم از حضور در سرویسهای راه دور منع شده‌اند (البته آنها می‌توانند هر کار دیگری انجام دهند، مثلاً چیپس و پفک بفروشند). در سال ۱۹۸۴ این دستور نسبتاً واضح بود، ولی تکنولوژی همواره راههای جالبی برای منسوخ کردن قوانین پیدا می‌کند. از آنجائیکه تلفنهای همراه و تلویزیون کابلی مشمول دستور MFJ نمی‌شوند، LEC ها و IXC ها بسمت خرید این شرکتها (یا ادغام با آنها) روی آوردند.

در سال ۱۹۹۵ کنگره متوجه شد که جدا نگه داشتن حوزهٔ فعالیت شرکتهای مختلف دیگر عملی نیست، و بهمین دلیل با تصویب یک لایحه به شرکتهای تلفن شهری، راه دور و تلویزیون کابلی اجازه داد تا وارد بازارهای یکدیگر شوند. ایدهٔ اصلی این لایحه اینست که سرویسهای صدا، داده و تلویزیون کابلی را یکپارچه کرده، و باعث رقابت بین شرکتهای مختلف (برای سرویس بهتر و قیمت کمتر) شود. این قانون از فوریه ۱۹۹۶ به اجرا گذاشته شد، و باعث شد تا تعدادی از BOC ها وارد حوزهٔ IXC شوند، و از طرف دیگر شرکتهای تلویزیون کابلی نیز به ارائهٔ سرویسهای تلفن بپردازند (و با LEC ها رقابت کنند).

یکی از نکات جالب قانون ۱۹۹۶ آنست که LEC ها بایستی شماره‌های تلفن را بگونه‌ای تنظیم کنند که قابل‌انتقال باشند. این بدان معناست که یک مشترک می‌تواند از این منطقه به منطقهٔ دیگر برود، بدون آنکه نیازی باشد شماره تلفنش را عوض کند. این ویژگی باعث می‌شود تا افراد آزادی بیشتری در عوض کردن LEC خود داشته باشند، و در نتیجه تنور رقابت داغتر شود. با قانون ۱۹۹۶، مخابرات ایالات متحده دستخوش تغییرات ساختاری شدیدی شده است. کشورهای بسیاری نیز ترغیب شده‌اند تا دست به چنین اقداماتی بزنند. البته در اغلب

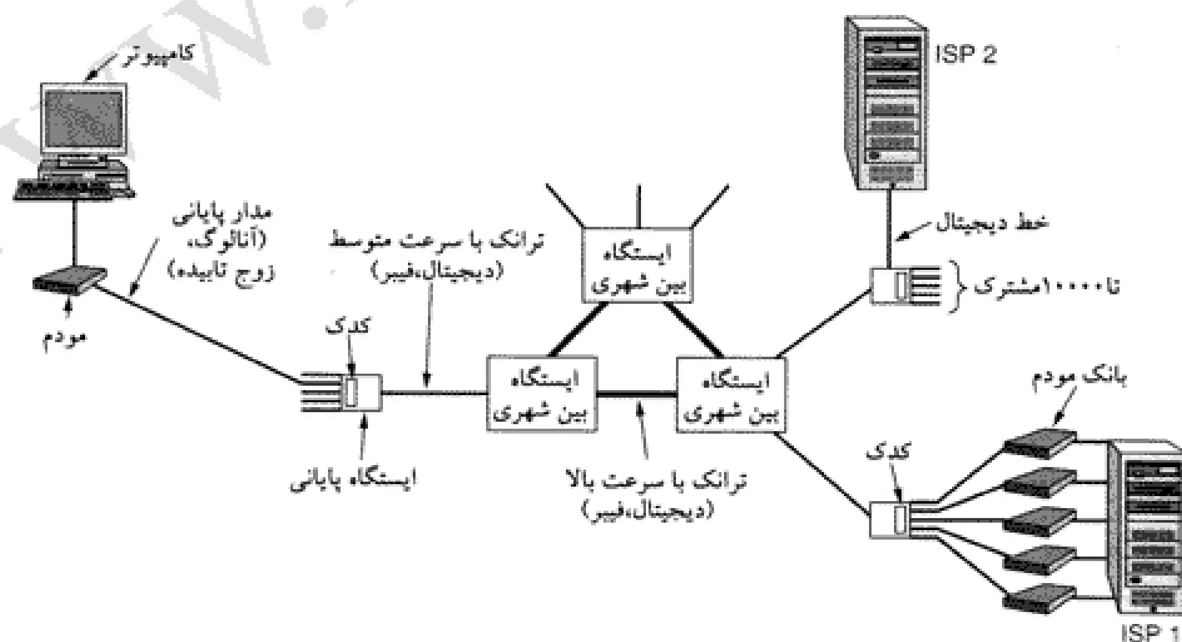
موارد کشورهای دیگر صبر می‌کنند تا نتیجه کار در ایالات متحده مشخص شود: اگر نتیجه مثبت بود، همان راه را می‌روند؛ اگر نتیجه منفی بود، راه دیگری را امتحان می‌کنند.

۲-۵-۳ مدارهای پایانی: مودم، ADSL، و بیسیم

اکنون زمان آن است که تا به بررسی نحوهٔ کار سیستم تلفن بپردازیم. بخشهای اصلی این سیستم در شکل ۲-۲۳ نشان داده شده است. در این شکل مدارهای پایانی، ترانک‌ها، ایستگاه‌های بین‌شهری و ایستگاههای پایانی را می‌بینید. هر ایستگاه پایانی (در ایالات متحده و کشورهای بزرگ) تا ۱۰,۰۰۰ مدار پایانی دارد. در حقیقت تا همین اواخر، **کُد منطقه + شمارهٔ ناحیه** مشخص کنندهٔ ایستگاه پایانی بود - برای مثال، 601-xxxx (212) یک ایستگاه پایانی با ۱۰,۰۰۰ مشترک (از 0000 تا 9999) است. با ایجاد شدن امکان رقابت بر سر سرویسهای محلی دیگر این وضعیت عملی نیست، چون شرکتهای زیادی طالب بدست آوردن کُدهای محلی هستند. روشهای شماره‌گذاری نیز بایستی بکلی عوض شود، چون اغلب کُدهای منطقه‌ای مصرف شده‌اند.

اجازه دهید با بخشی که بیشتر مردم با آن آشنایند، شروع کنیم: دو رشته سیمی که از ایستگاه پایانی شرکت تلفن به محل مشترک (خانه یا محل کار) کشیده می‌شود. به این دو رشته سیم اغلب «**کیلومتر آخر**» گفته می‌شود، اگرچه طول آن ممکنست به چندین کیلومتر هم برسد. در طول یکصد سال گذشته در این بخش از سیگنالهای آنالوگ استفاده شده است، و (بدلیل هزینهٔ زیاد تجهیزات دیجیتال) احتمالاً در چند سال آینده نیز وضع بهمین منوال خواهد بود. با این حال، در این آخرین سنگر آنالوگ نیز تغییرات شروع شده است. در این قسمت مدارهای پایانی و آخرین تحولات آنرا (با تأکید بر مخابرات داده بین کامپیوترها) بتفصیل بررسی خواهیم کرد.

وقتی یک کامپیوتر می‌خواهد داده‌های دیجیتال را روی یک خط تلفن آنالوگ ارسال کند، ابتدا باید آنها را به سیگنالهای آنالوگ تبدیل کند - کاری که با دستگاهی بنام **مودم** (modem) انجام می‌شود. در ایستگاه پایانی این اطلاعات مجدداً به سیگنالهای دیجیتال تبدیل شده، و برای ارسال روی ترانک راه دور (trunk) آماده می‌شود. اگر مقصد اطلاعات یک کامپیوتر باشد، سیگنال مجدداً به آنالوگ تبدیل می‌شود تا بتوان آنرا روی مدار پایانی



شکل ۲-۲۳. ترکیبی از انتقال دیجیتال و آنالوگ برای تماس کامپیوتر با کامپیوتر. تبدیل سیگنال بوسیلهٔ مودمها و کودکها انجام می‌شود.

آن ارسال کرد؛ و در مقصد یک مودم دیگر اطلاعات را از آنالوگ به دیجیتال تبدیل می‌کند. در شکل ۲-۲۳، ISP 1 (ارائه‌دهندهٔ سرویسهای اینترنتی) یک بانک مودم دارد، که هر کدام از آنها به یک خط تلفن (مدار پایانی) مستقل متصلند. این ISP می‌تواند در آن واحد به تعداد مودمهای خود به افراد مختلف سرویس بدهد (البته با این فرض که کامپیوترهایش به اندازهٔ کافی قوی باشند). این آرایش تا وقتی که مودمهای 56-kbps وارد بازار شدند، آرایش متداولی بود (بزودی علت آنرا خواهید فهمید).

در مخابرات آنالوگ برای انتقال اطلاعات از ولتاژی که با زمان تغییر می‌کند، استفاده می‌شود. اگر رسانهٔ انتقال کامل و بدون نقص باشد، گیرنده دقیقاً همان سیگنالی را دریافت خواهد کرد که فرستنده ارسال کرده است. اما متأسفانه چنین نیست، و گیرنده همان سیگنال ارسالی را دریافت نمی‌کند - و در مخابرات دیجیتال این یعنی خطا.

انتقال سیگنالهای الکتریکی روی خطوط انتقال با سه مشکل عمده روبروست: تضعیف سیگنال، اعوجاج تأخیری، و نویز. وقتی یک سیگنال در رسانهٔ انتقال منتشر می‌شود، انرژی خود را از دست می‌دهد که به آن **تضعیف** (attenuation) می‌گویند، و بر حسب دسی‌بل بر کیلومتر اندازه‌گیری می‌شود. میزان تضعیف یک سیگنال به فرکانس آن بستگی دارد. شاید فکر کنید که این وابستگی به فرکانس اشکال زیادی ایجاد نمی‌کند، ولی وقتی یک موج را بصورت مجموعه‌ای از مؤلفه‌های فوریه در نظر بگیرید، تأثیر وابستگی به فرکانس خود را نشان خواهد داد. در واقع هر یک از مؤلفه‌های فوریه بگونه‌ای متفاوت تضعیف می‌شوند، و ترکیب مجدد آنها در گیرنده موج کاملاً متفاوتی ایجاد می‌کند.

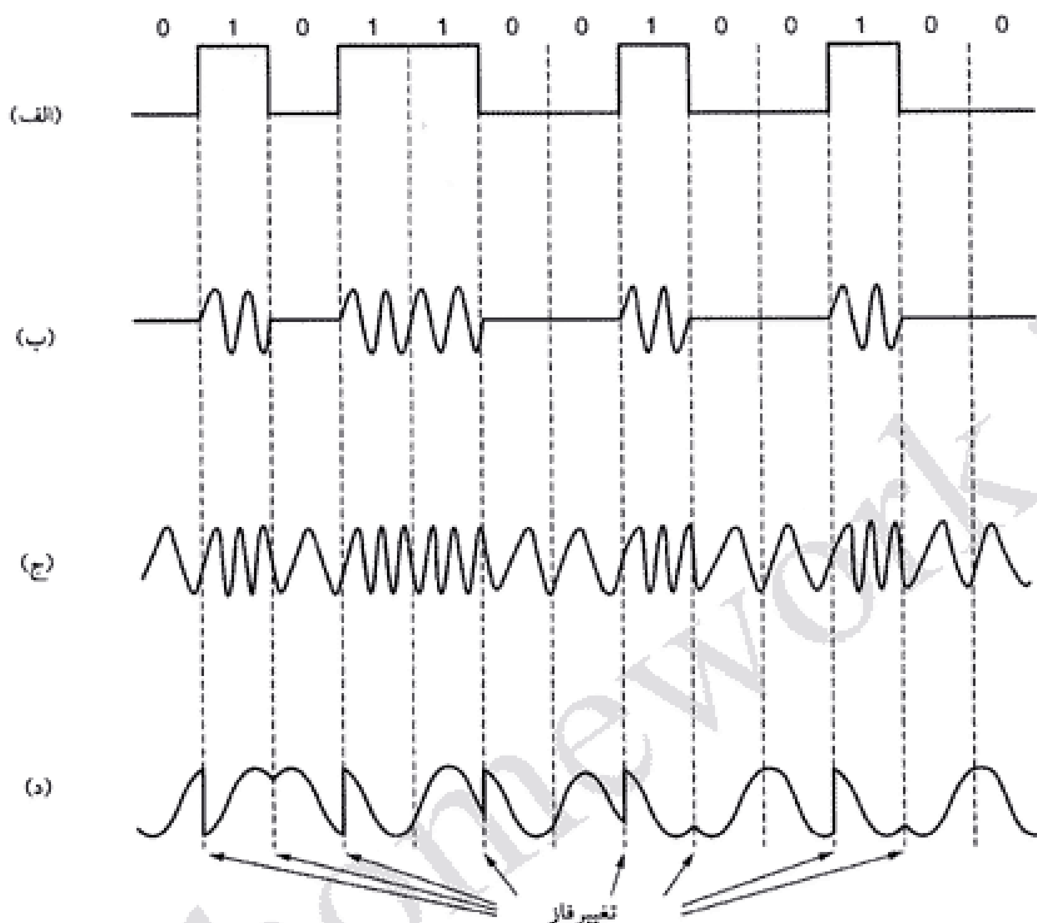
اما وضع از این هم بدتر است، چون مؤلفه‌های فوریه با سرعتهای متفاوتی در رسانهٔ انتقال (سیم) منتشر می‌شوند. این تفاوت سرعتها باعث **اعوجاج تأخیری** (delay distortion) سیگنال در گیرنده می‌شود.

مشکل دیگر **نویز** (noise) - انرژی ناخواسته از منابعی غیر از فرستنده - است. نویز حرارتی حاصل حرکات تصادفی الکترونها در سیم است، و بکلی نمی‌توان از آن اجتناب کرد. نویز القایی نیز حاصل القای ولتاژ در اثر عبور جریان از سیمهای مجاور است. گاهی پیش آمده که وقتی تلفنی با کسی صحبت می‌کنید، مکالمهٔ دیگری را نیز در زمینه می‌شنوید؛ علت این پدیده (که به **همشنوایی** - crosstalk - مشهور است) نویز القایی می‌باشد. در اثر قطع و وصل خطوط قدرت نویز دیگری بنام **نویز ضربه‌ای** روی خطوط مخابراتی القا می‌شود، که می‌تواند چندین بیت از اطلاعات را از بین ببرد.

مودم

بدلیل مشکلاتی که در بالا گفته شد (بویژه وابستگی تضعیف سیگنال و سرعت انتشار آن به فرکانس)، سعی می‌شود از سیگنالهایی با محدودهٔ فرکانسی پائین استفاده شود. متأسفانه، شکل موج مربعی سیگنالهای دیجیتال دارای طیف فرکانسی وسیعی است، و بشدت در معرض تضعیف و اعوجاج تأخیری قرار دارد. این تأثیرات باعث شده تا سیگنالهای بیس‌باند (DC) فقط برای سرعتهای پائین و مسافتهای کوتاه مناسب باشد.

برای حل مشکل سیگنالهای DC بویژه در خطوط تلفن، از سیگنال AC استفاده می‌شود. در این جا یک تون ۱۰۰۰ تا ۲۰۰۰ هرتزی بعنوان **موج حامل سینوسی** (sine wave carrier) بکار برده می‌شود. برای انتقال اطلاعات می‌توان دامنه، فرکانس یا فاز این موج حامل را مدوله کرد. در **مدولاسیون دامنه** (amplitude modulation) از دو دامنهٔ متفاوت بعنوان 0 و 1 استفاده می‌شود. در **مدولاسیون فرکانس** (frequency modulation) -که به **کُدگذاری با شیفت فرکانس** (frequency shift keying) نیز معروف است- از دو تون متفاوت برای 0 و 1 استفاده می‌شود. (در صنعت مخابرات، اصطلاحات مدولاسیون و کُدگذاری معادل یکدیگرند.) در **مدولاسیون فاز** (phase modulation) موج حامل در فواصل یکنواخت 0 یا 180 درجه شیفت پیدا می‌کند. با استفاده از شیفتهای 45 ، 135 ، 225 یا 315 درجه‌ای می‌توان در هر فاصلهٔ زمانی بجای یک بیت، ۲



شکل ۲-۲۴. (الف) سیگنال باینری. (ب) مدولاسیون دامنه. (ج) مدولاسیون فرکانس. (د) مدولاسیون فاز.

بیت اطلاعات منتقل کرد. همچنین، وجود تغییر فاز در انتهای هر فاصلهٔ زمانی تشخیص مرزهای آنها را برای گیرنده آسانتر می‌کند.

شکل ۲-۲۴ این سه نوع مدولاسیون را نشان می‌دهد. در شکل ۲-۲۴ (ب) یکی از دامنه‌ها صفر و دیگری غیرصفر است. در شکل ۲-۲۴ (ج) از دو فرکانس متفاوت برای نمایش 0 و 1 استفاده شده است. در شکل ۲-۲۴(د) در مرز هر فاصلهٔ زمانی وجود (یا عدم وجود) تغییر فاز نشاندهندهٔ تغییر مقدار بیت (یا عدم تغییر آن) است.

دستگاهی که جریان بیت‌ها را بعنوان ورودی گرفته، و با ایجاد یک موج حامل و اعمال یکی از انواع مدولاسیون (یا ترکیبی از آنها) یک خروجی آنالوگ تولید می‌کند (و یا بر عکس، با گرفتن موج آنالوگ اطلاعات دیجیتال را از آن استخراج می‌کند)، **مودم** (modulator-demodulator) نامیده می‌شود. مودم بین کامپیوتر (دیجیتال) و سیستم تلفن (آنالوگ) قرار می‌گیرد.

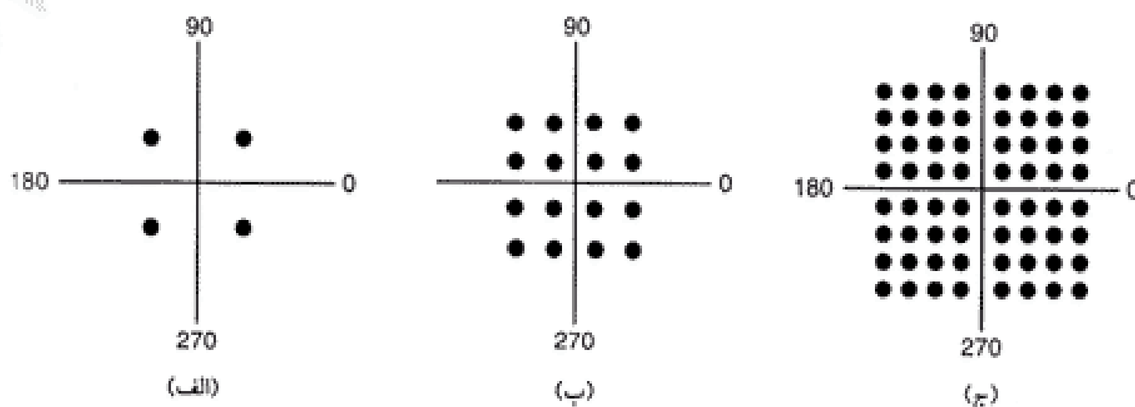
بالا بردن سرعت بسادگی و فقط با زیاد کردن نرخ نمونه‌برداری (sampling rate) ممکن نیست. قضیهٔ نایکوئیست می‌گوید که برای یک خط کامل 3000 Hz (که خطوط تلفن مسلماً چنین نیستند)، حداکثر نرخ نمونه‌برداری 6000 Hz است. در عمل، اکثر مودمها با نرخ 2400 times/sec نمونه‌برداری می‌کنند، ولی سعی می‌کنند در هر نمونه‌برداری بیت‌های بیشتر را بخوانند.

به تعداد نمونه‌ها در ثانیه **باد** (baud) گفته می‌شود، و در هر باد یک **سمبل** (symbol) فرستاده می‌شود. بنابراین، یک خط n-baud در هر ثانیه n سمبل ارسال می‌کند (برای مثال، یک خط 2400-baud در هر 416.667 μsec یک سمبل می‌فرستد). اگر این سمبل فقط حاوی ولتاژ 0 v (برای نمایش 0 منطقی) یا 1 v (برای نمایش 1 منطقی) باشد، سرعت مودم 2400 bps خواهد بود. اما اگر از ولتاژهای 0 ، 1 ، 2 و 3 ولت استفاده کنیم، هر سمبل می‌تواند حاوی ۲ بیت باشد، و سرعت انتقال اطلاعات مودم به 4800 bps می‌رسد. در مدولاسیون فاز چهار درجه‌ای نیز می‌توان در هر سمبل ۲ بیت ارسال کرد، و بدین ترتیب سرعت ارسال داده دو برابر سرعت باد خواهد بود. این تکنیک که کاربرد زیادی گسترده‌ای نیز دارد، QPSK (**کُدگذاری با شیفت فاز چهارگانه** – Quadrature Phase Shift Keying) نامیده می‌شود.

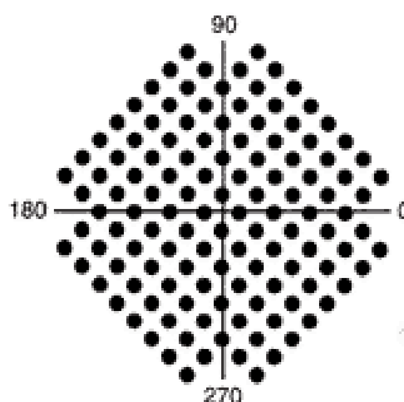
مفاهیم پهنای باند، باد، سمبل، و نرخ بیت بسیار با هم اشتباه می‌شوند، بنابراین اجازه دهید یک بار دیگر آنها را بیان کنیم. **پهنای باند** یک رسانه محدودهٔ فرکانسی است که می‌تواند سیگنال را با کمترین تضعیف عبور دهد. این یکی از ویژگیهای فیزیکی رسانهٔ انتقال است، و با هرتز (Hz) سنجیده می‌شود. به تعداد نمونه‌برداری در هر ثانیه **باد** گفته می‌شود، و هر نمونه حاوی یک قطعه از اطلاعات (یا **سمبل**) است. بنابراین، **نرخ باد** (baud rate) و **نرخ سمبل** (symbol rate) در واقع یکی هستند. تعداد بیت بر سمبل توسط نوع مدولاسیون (مثلاً QPSK) تعیین می‌شود. **نرخ بیت** (bit rate) مقدار اطلاعاتیست که روی یک کانال فرستاده می‌شود، و برابر است با نرخ سمبل (symbol/sec) × تعداد بیت در هر سمبل (bits/symbol) .

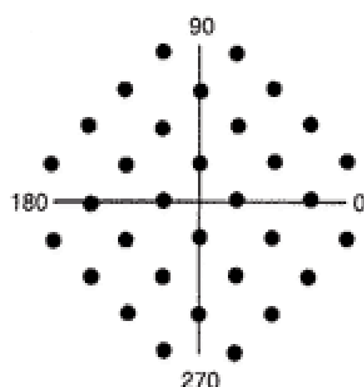
تمام مودمهای پیشرفته برای ارسال بیشترین بیتهای ممکن در هر باد، از مدولاسیونهای ترکیبی (چند دامنه‌ای و چند فازی) استفاده می‌کنند. در شکل ۲-۲۵ (الف) نقاطی را می‌بینید که در فواصل یکسان از مبدأ مختصات (دامنهٔ یکسان)، و با زاویه‌ها (فازها) ی 45 ، 135 ، 225 و 315 درجه قرار گرفته‌اند (فاز هر نقطه زاویه‌ایست که با جهت مثبت محور x می‌سازد). در شکل ۲-۲۵ (الف) چهار ترکیب معتبر وجود دارد، که بدین ترتیب می‌توان در هر سمبل ۲ بیت ارسال کرد – این همان QPSK است.

در شکل ۲-۲۵ (ب) مدولاسیون دیگری را می‌بینید، که در آن از چهار دامنه و چهار فاز مختلف استفاده شده و ۱۶ ترکیب معتبر بدست می‌دهد. با این مدولاسیون می‌توان در هر سمبل چهار بیت ارسال کرد، و به آن QAM-16 (**مدولاسیون دامنهٔ چهارگانه** – Quadrature Amplitude Modulation) گفته می‌شود (گاهی نیز 16-QAM خوانده می‌شود). با مدولاسیون QAM-16 می‌توان روی یک خط 2400-baud تا سرعت 9600 bps داده ارسال کرد.



شکل ۲-۲۵. (الف) مدولاسیون QPSK . (ب) مدولاسیون QAM-16 . (ج) مدولاسیون QAM-64 .





شکل ۲-۲۶. (الف) استاندارد V.32 برای 9600 bps . (ب) استاندارد V.32 bis برای 14,400 bps .

در شکل ۲-۲۵ (د) مدولاسیون دیگری را می‌بینید که شامل دامنه‌های بیشتری است. در این طرح ۶۴ ترکیب ممکنه وجود دارد، که بدین ترتیب می‌توان به ازای هر سمبل ۶ بیت را ارسال کرد. این مدولاسیون QAM-64 نام دارد (از مدولاسیون QAM با مراتب بالاتر نیز استفاده می‌شود).

دیاگرامهایی مانند شکل ۲-۲۵ که ترکیبات ممکنهٔ دامنه و فاز را نشان می‌دهند، به **دیاگرام فلکی** (constellation diagram) معروفند. هر استاندارد مودمهای سرعت-بالا دارای دیاگرام فلکی خاص خود است، و فقط قادر به ارتباط با مودمهائیست که دارای دیاگرام مشابهی باشند (البته این مودمها می‌توانند تمام سرعتهای پائینتر را شبیه‌سازی کنند).

با بالا رفتن تعداد نقاط در دیاگرام فلکی حتی نویزهای کوچک نیز می‌توانند باعث بروز خطا در آشکارسازی دامنه یا فاز سیگنال شوند، که بدنبال آن بیتهای زیادی از دست می‌رود. برای کاهش احتمال خطا، در مودمهای سرعت-بالا با اضافه کردن بیت‌های اضافی نوعی **تصحیح خطا** (error correction) انجام می‌شود. به این روش TCM (**مدولاسیون کُدگذاری تاروپودی** - Trellis Coded Modulation) می‌گویند. برای مثال، استاندارد V.32 از ۳۲ نقطهٔ فلکی برای ارسال ۴ بیت داده و یک بیت **برابری** (parity) در هر سمبل استفاده کرده، و با نرخ 2400-baud به سرعت 9600 bps (با تصحیح خطا) دست می‌یابد. دیاگرام فلکی این استاندارد را در شکل ۲-۲۶ (الف) مشاهده می‌کنید. (چرخش ۴۵ درجه‌ای حول محور مختصات فقط بدلایل مهندسی اتخاذ شده است، و هیچگونه تأثیری روی ظرفیت اطلاعات ندارد.)

بعد از 9600 bps قدم بعدی 14,400 bps است، که استاندارد آن V.32 bis نامیده می‌شود. برای رسیدن به این سرعت باید (با نرخ 2400-baud) در هر سمبل ۶ بیت داده و ۱ بیت برابری ارسال شود. دیاگرام فلکی این مودم را، که (وقتی از QAM-128 استفاده شود) ۱۲۸ نقطه دارد، در شکل ۲-۲۶ (ب) می‌بینید. مودمهایی که قابلیت فکس دارند، برای ارسال فکس از این استاندارد استفاده می‌کنند. مدولاسیون QAM-256 در هیچ یک از مودمهای تلفنی استفاده نمی‌شود، ولی در شبکه‌های کابلی کاربرد دارد (بعداً آنرا خواهید دید).

استاندارد مودم تلفنی بعدی V.34 است، که با سرعت 28,800 bps (12 data bits/symbol در 2400-baud) کار می‌کند. آخرین مودم در این سری استاندارد V.34 bis نام دارد، که به سرعت 33,600 bps (14 data bits/symbol در 2400-baud) دست پیدا می‌کند.

برای رسیدن به سرعتهای بیشتر، بسیاری از مودمها از تکنیکهای فشرده‌سازی استفاده می‌کنند تا به سرعتهای

بالاتر از 33,600 bps برسند. از سوی دیگر، تقریباً تمام مودمها قبل از شروع ارسال داده‌ها کیفیت خط را چک می‌کنند، و اگر نقصی در کیفیت خط وجود داشته باشد، سرعت خود را آنقدر پائین می‌آورند تا ارسال مطمئن داده‌ها امکانپذیرباشد. بهمین دلیل، سرعت مؤثر یک مودم می‌تواند کمتر، مساوی، و یا بیشتر از سرعت رسمی آن باشد.

تمام مودمهای جدید (با استفاده از فرکانسهای متفاوت برای ارسال و دریافت) اجازه می‌دهند تا ارسال و دریافت همزمان انجام شود. به چنین ارتباطی **دو-طرفهٔ همزمان** (full duplex) گفته می‌شود، مانند یک اتوبان دو-بانده. اگر در هر لحظه فقط از یک طرف ارتباط ممکن باشد، به آن **دو-طرفهٔ ناهمزمان** (half duplex) گفته می‌شود (مانند راه‌آهن‌های یک-خطه). و اگر ارتباط فقط در یک جهت مجاز باشد، بـه آن **یکـطرفه** (simplex) می‌گویند (مانند یک خیابان یکطرفه). یک رشته فیبر نوری که در یک سمت فقط دیود لیزری و در سمت دیگر فقط آشکارساز نوری دارد، نیز سیستمی یکطرفه است.

همانطور که قبلاً گفتیم، طبق قانون شانون سرعت انتقال روی خطوط تلفن از 35-kbps نمی‌تواند فراتر رود، بهمین مودمهای استاندارد از سرعت 33,600 bps بالاتر نمی‌روند. شاید بپرسید پس مودمهای 56-kbps چطور کار می‌کنند؟ کمی صبر کنید، به آن هم خواهیم رسید.

اما این حد 35-kbps از کجا آمده است؟ این محدودیت به طول متوسط مدارهای پایانی و کیفیت آنها بستگی دارد. به شکل ۲-۲۳ نگاه کنید: ارتباطی که از کامپیوتر سمت چپ شروع شده و به ISP 1 ختم شود، از دو مدار پایانی آنالوگ (یکی در مبدأ و دیگری در مقصد) عبور می‌کند. هر یک از این حلقه‌ها میزان نویز سیگنال را بالا می‌برند. اگر بتوانیم یکی از این حلقه‌ها را حذف کنیم، می‌توانیم حداکثر سرعت را به دو برابر برسانیم.

این دقیقاً همان کاریست که ISP 2 انجام داده: این ISP ارتباط خود با نزدیکترین ایستگاه پایانی را بصورت کاملاً دیجیتال در آورده است. سیگنال دیجیتال ترانک مستقیماً به ISP 2 فرستاده شده، و مودمها و کـودکهای آنالوگ بکلی حذف شده‌اند. بدین ترتیب، حداکثر سرعت ارتباط با این ISP می‌تواند به 70-kbps نیز برسد. بین دو نقطه که از مودم و خطوط آنالوگ استفاده می‌کنند، حداکثر سرعت همان 33.6 kbps است.

اما علت اینکه مودمهای 56-kbps می‌توانند در عمل کار کنند، به قضیهٔ نایکوئیست برمی‌گردد. پهنای بـاند کانالهای تلفنی (بانضمام باند محافظ) 4000 Hz است، و طبق قضیهٔ نایکوئیست حداکثر نرخ نمونه‌برداری در چنین کانالی 8000 خواهد بود. در ایالات متحدهٔ آمریکا تعداد بیتهای هر نمونه ۸ است، که (با احتساب یک بیت کنترلی) حداکثر سرعت مجاز به 56,000 bits/sec می‌رسد. در اروپا از بیت کنترلی استفاده نمی‌شود، و می‌توان تمام بیتها را به داده‌ها اختصاص داد، بنابراین حداکثر سرعت در آنجا می‌تواند تا 64,000 bits/sec افزایش یابد، ولی طبق یک توافق بین‌المللی همان سرعت 56,000 انتخاب شده است.

این استاندارد V.90 نامیده می‌شود. در این استاندارد سرعت ارسال کاربر به ISP همان 33.6 kbps است، ولی سرعت دریافت از ISP به 56-kbps می‌رسد (چون معمولاً آنچه که کاربر از ISP مـی‌گیرد، بسیار بیشتر از آن چیزیست که به آن می‌فرستند). از لحاظ نظری امکان رساندن کانال ارسال به ISP تا 56-kbps نیز وجود داشت، ولی (بعلت نویزی بودن خطوط تلفنی) تصمیم گرفته شد تا این کانال به 33.6 kbps محدود شده، و پهنای باند آن به کانال دریافت از ISP داده شود تا احتمال رسیدن آن به سرعت 56-kbps افزایش یابد.

قدم بعدی در این بازی استانداردها، V.92 است. مودمهای این استاندارد می‌توانند تا 48-kbps روی کانال ارسال داده بفرستند (البته مشروط باینکه خط تلفن توانایی آنرا داشته باشد). زمان شناسایی کیفیت خط و تعیین سرعت مناسب در این استاندارد نیز بسیار کمتر از مودمهای دیگر است (تقریباً نصف ۳۰ثانیه‌ای که مودمهای دیگر صرف این کار می‌کنند). و بالاخره، مودمهای استاندارد V.92 اجازه می‌دهند تا یک تماس تلفنی بتوانـد تـماس اینترنتی را قطع کند، مشروط باینکه خط دارای سرویس **انتظار مکالمه** (call waiting) باشد.

خط مشترک دیجیتال (DSL)

وقتی شرکتهای تلفن بالاخره موفق شدند سرویس 56-kbps ارائه کنند، خیلی به خود غره شدند؛ اما در همان حال، شرکتهای تلویزیون کابلی ارتباطاتی با سرعت 10 Mbps (و ماهواره‌ها سرعت ارسال 50 Mbps) عرضه می‌کردند. با داغ شدن بازار اینترنت، شرکتهای تلفن (LEC) دریافتند که برای رقابت به محصول جدیدی نیاز دارند. پاسخ آنها به این وضعیت ارائهٔ سرویسهای دیجیتال روی مدارهای پایانی بود. این سرویسها پهنای باند بیشتری داشتند و به آنها سرویس **باند-وسیع** (broadband) گفته می‌شد (اگرچه این نامگذاری بیشتر جنبهٔ تبلیغاتی داشت، تا فنی).

در ابتدا این سرویسها بسیار متنوع بودند، و تحت نام xDSL (**خط مشترک دیجیتال**: Digital Subscriber Line – که در آن x نوع سرویس را مشخص می‌کند) دسته بندی می‌شدند. مهمترین این سرویسها – که عامل اصلی موفقیت آن هم بود – ADSL (DSL **نامتقارن** – Asymmetric DSL) نام دارد، که در زیر با آن آشنا خواهید شد. از آنجائیکه ADSL هنوز در حال توسعه و تکامل است و استانداردهای آن هنوز بطور کامل تدوین نشده، ممکنست در آینده تغییراتی در آن رخ دهد، ولی تصویر کلی همان است که خواهید دید. برای اطلاعات بیشتر دربارهٔ ADSL به (Summers, 1999; Vetter et al., 2000) نگاه کنید.

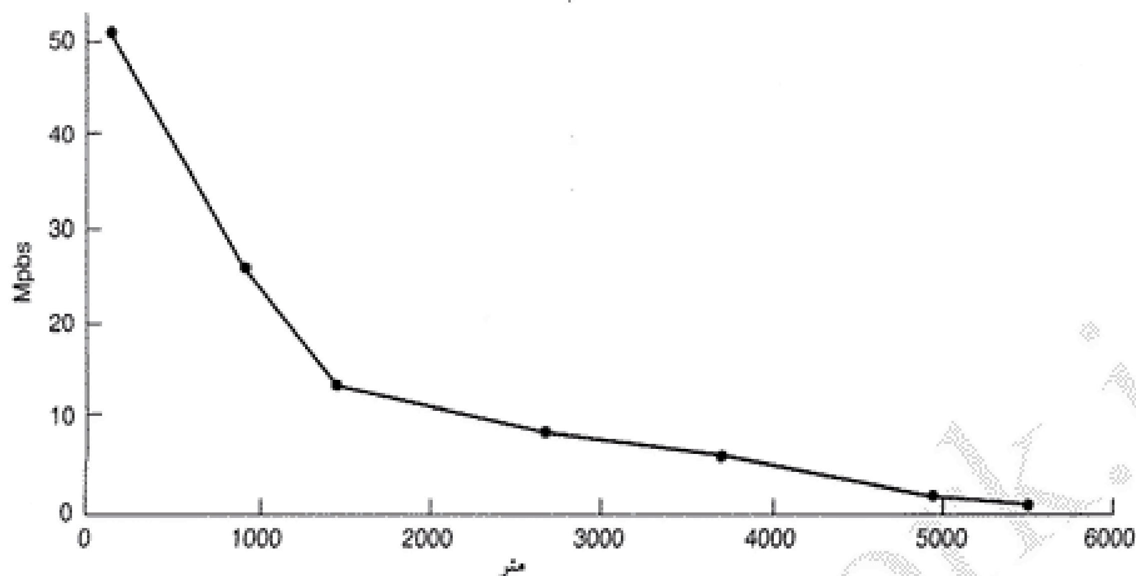
علت آن که مودمها اینقدر کُند هستند اینست که، شبکهٔ تلفن اساساً برای انتقال صدای انسان طراحی و توسعه داده شده، و سرویسهای داده فرزندخواندهٔ آن محسوب می‌شود. در نقطه‌ای که م.ار پایانی وارد ایستگاه پایانی تلفن می‌شود، فیلترهایی قرار داده شده که تمام فرکانسهای زیر 300 Hz و بالای 3400 Hz را تضعیف می‌کنند. البته قطع فرکانس بیکباره صورت نمی‌گیرد – از لحاظ فنی، 300 Hz و 3400 Hz نقاط 3 dB هستند – بهمین دلیل پهنای باند را معمولاً 4000 Hz فرض می‌کنند، اگرچه فاصلهٔ نقاط 3 dB فقط 3100 Hz است. داده نیز به همین باند باریک محدود است.

برای اجتناب از این وضعیت در سرویس xDSL ، خط مشترک بدون عبور از فیلترهای مزبور مستقیماً به نوع خاصی از سوئیچها متصل می‌شود، تابتواند از تمام ظرفیت مدار پایانی استفاده کند. در این حالت دیگر محدودیت پهنای باند فقط به خواص فیزیکی مدار پایانی بستگی دارد، نه به محدوده‌ای که فیلترها بطور مصنوعی برای آن بوجود آورده‌اند.

متأسفانه، ظرفیت مدار پایانی نامحدود نیست، و به عواملی از قبیل طول خط، ضخامت سیم، و کیفیت کلی آن بستگی دارد. در شکل ۲-۲۷ نمودار پهنای باند بر حسب مسافت را ملاحظه می‌کنید – در اینجا فرض شده که سایر عوامل بهینه هستند (سیمهای نو، کابلهای نه چندان قطور، و غیره).

نمودار فوق مشکل اصلی شرکتهای تلفن را بخوبی نشان می‌دهد: شعاع ارائهٔ این سرویس به مشترکان بشدت محدود است. این بدان معناست که وقتی کاربری که خارج از این شعاع زندگی می‌کند، برای دریافت سرویس xDSL مراجعه کند، باید با کمال تأسف از او عذرخواهی کنیم که امکان ارائهٔ سرویس به وی را نداریم. برای بیشتر کردن شعاع سرویس، باید سرعت آنرا پائین بیاوریم، ولی پائین آوردن سرعت همان و از دست دادن جذابیت همان. اینجاست که تکنولوژی مغلوب اقتصاد می‌شود. (راه حل دیگر آنست که ایستگاه‌های کوچک و پراکنده‌ای در نقاط نزدیک به هم تأسیس کنیم، که البته این هم اقتصادی نیست.)

سرویس xDSL با اهداف مشخصی طراحی شده است: اول اینکه، این سرویسها باید بتوانند با خطوط زوج تابیده Cat 3 کار کنند؛ دوم اینکه، این سرویسها نباید هیچ اختلالی در دستگاههای تلفن و فکس معمولی بوجود آورند؛ سوم اینکه، باید از 56-kbps سریعتر باشند؛ و چهارم اینکه، این سرویسها باید دائماً برقرار باشند، و هزینهٔ آنها هم ثابت (و مثلاً ماهیانه) باشد – نه مانند تلفنهای معمولی، بصورت دقیقه‌ای.

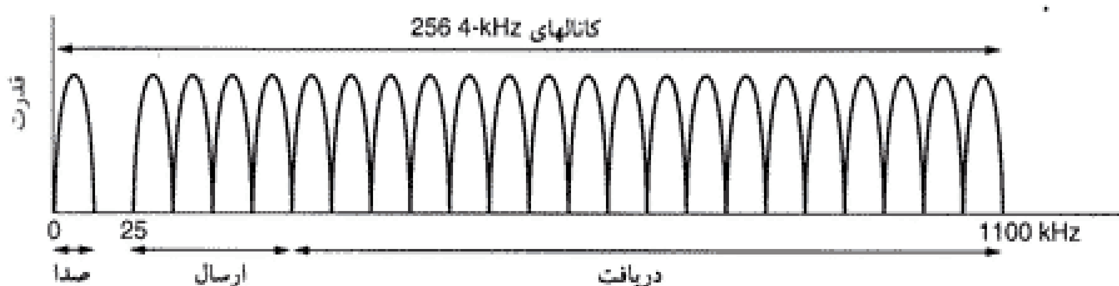


شکل ۲-۲۷. نمودار پهنای باند سرویس DSL بر حسب طول سیم در کابل Cat 3 UTP.

اولین سرویس ADSL توسط AT&T ارائه شد، که در آن پهنای باند موجود در مدار پایانی (که تقریباً 1.1 MHz است) به سه باند تقسیم شده بود: باند POTS (**سرویس تلفن معمولی** – Plain Old Telephone Service)، باند ارسال از کاربر (به ایستگاه پایانی)، و باند ارسال به کاربر (از ایستگاه پایانی). تکنیک تقسیم پهنای باند به فرکانسهای مختلف **مالتی‌پلکس تقسیم فرکانس** (frequency division multiplexing) نام دارد، که بعداً دربارهٔ آن مفصلاً صحبت خواهیم کرد. شرکتهای بعدی برای ارائهٔ این سرویس از تکنیک متفاوتی استفاده کردند، و از آنجائیکه احتمالاً این روش غالب خواهد شد، ابتدا آنرا توضیح می‌دهیم.

این روش که DMT (**تون چندگانهٔ گسسته** – Discrete MultiTone) نامیده می‌شود، در شکل ۲-۲۸ نشان داده شده است. در حقیقت، کاری که در اینجا انجام شده تقسیم پهنای باند موجود (1.1 MHz) به 256 کانال مستقل 4312.5 Hz است. از کانال 0 بعنوان POTS (سرویس تلفن معمولی) استفاده می‌شود. کانالهای 1-5 خالی رها شده‌اند، تا تداخلی بین صدا و داده پیش نیاید. از 250 کانال باقیمانده، یکی برای کنترل ارسال از کاربر و یکی برای کنترل ارسال به کاربر تخصیص یافته، و از بقیهٔ کانالها می‌توان برای داده استفاده کرد.

در تئوری می‌توان از هر یک از این کانالها برای ارتباط دو-طرفهٔ همزمان استفاده کرد، ولی هارمونیها، همشنوایی و اثرات دیگر باعث می‌شود تا در عمل ظرفیت سیستم بسیار کمتر باشد. این ارائه‌دهندهٔ سرویس است که تعیین می‌کند چند کانال برای ارسال از کاربر اختصاص یافته، و چند کانال برای ارسال به کاربر. از نظر تکنیکی می‌توان این نسبت را بصورت 50-50 تعریف کرد، ولی از آنجائیکه اغلب کاربران اطلاعات دریافتی خیلی بیشتری دارند، ۸۰ الی ۹۰ درصد پهنای باند به دریافت کاربر اختصاص داده می‌شود. از اینجاست که حرف "A" (بمعنای



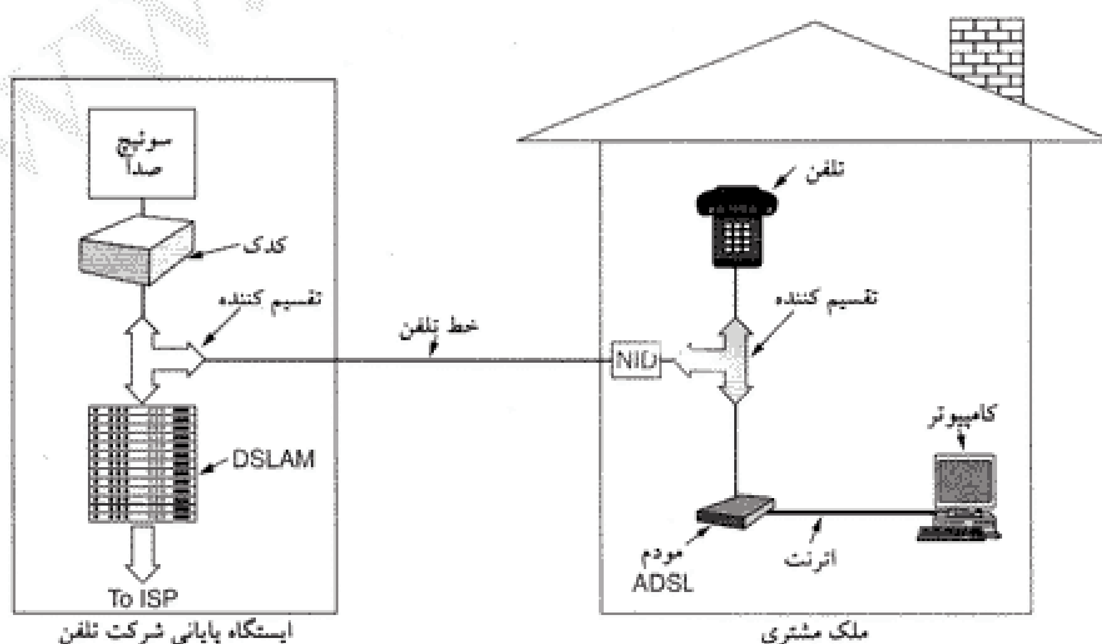
شکل ۲-۲۸. عملکرد ADSL با مدولاسیون تون چندگانهٔ گسسته.

"نامتقارن") در ADSL ظاهر می‌شود. در اغلب موارد ۳۲ کانال به ارسال کاربر، و بقیه به دریافت آن اختصاص می‌یابد. بمنظور افزایش پهنای باند، امکان اختصاص چند کانال فوقانی به ارتباط دوطرفه نیز وجود دارد، ولی این روش به تجیزات اضافی برای خنثی کردن **پژواک** (echo) در خط نیاز دارد.

در استاندارد ADSL (ANSI T1.413 و ITU G.992.1) تا 8 Mbps برای دریافت کاربر و تا 1 Mbps برای ارسال کاربر مجاز است. اما، کمتر شرکتی پیدا می‌کنید که چنین سرویسهایی ارائه کند. در عمل، این سرویسها معمولاً بصورت 512 kbps دریافت و 64 kbps ارسال (سرویس معمولی)، و 1 Mbps دریافت و 256 kbps ارسال (سرویس ویژه) ارائه می‌شوند.

در هر کانال از مدولاسیونی شبیه V.34 (با نرخ نمونه‌برداری 4000-baud بجای 2400-baud) استفاده می‌شود. کیفیت خط در هر کانال مستقلاً (و بصورت پیوسته) ارزیابی شده، و در صورت نیاز سرعت مجدداً تنظیم می‌شود، بنابراین سرعت کانالها می‌تواند کاملاً متفاوت باشد. برای ارسال داده‌ها از مدولاسیون QAM با نرخ 15 bits/baud (شبیه شکل ۲-۲۵ ب) استفاده می‌شود. برای مثال، اگر 224 کانال دریافت با نرخ 15 bits/baud در 4000-baud داشته باشیم، پهنای باند دریافتی معادل 13.44 Mbps خواهیم داشت. در عمل، نرخ سیگنال-به-نویز هرگز آنقدر خوب نیست که اجازهٔ چنین سرعتهای رابدهد، ولی در مسافتهای کوتاه و با کابل‌کشی مناسب می‌توان به سرعت 8 Mbps رسید، که همین برای استقبال عامه کافیست.

در شکل ۲-۲۹ یک طرح ADSL را مشاهده می‌کنید. در این طرح، تکنسین شرکت تلفن بایستی یک NID (**دستگاه واسط شبکه** - Network Interface Device) در محل سکونت مشتری نصب کند. این جعبهٔ پلاستیکی کوچک انتهای مایملک شرکت تلفن و شروع مایملک مشتری را مشخص می‌کند. کنار NID (یا حتی گاهی بصورت ترکیبی با آن) یک **تقسیم‌کننده** (splitter - فیلتر آنالوگی که باند POTS را از باند داده جدا می‌کند) نصب می‌شود. سیگنال POTS به دستگاه تلفن، و سیگنال داده به یک مودم ADSL داده می‌شود. مودم ADSL در واقع یک DSP (**پردازشگر سیگنال دیجیتال** - Digital Signal Processor) است، که بگونه‌ای تنظیم شده تا بعنوان 256 مودم QAM موازی (با فرکانسهای مختلف) کار کند. از آنجائیکه اکثر مودمهای ADSL بصورت خارج از



شکل ۲-۲۹. یک طرح سادهٔ ADSL .

کامپیوتر هستند، ارتباط آنها با کامپیوتر نیز باید از نوع پُرسرعت باشد. برای این منظور از پورتهای اترنت یا USB استفاده می‌شود. بدون شک در آینده شاهد به بازار آمدن مودمهای داخلی ADSL نیز خواهیم بود.

در انتهای دیگر خط (در ایستگاه پایانی)، یک تقسیم‌کنندهٔ متناظر قرار می‌گیرد. در اینجا، سیگنال صدا (0-4000 Hz) از داده جدا شده و به سوئیچ‌های معمولی تلفن فرستاده می‌شود. سیگنال بالای 26 kHz نیز جدا شده، و به دستگاهی بنام DSLAM (**مالتی‌پلکسر دسترسی** DSL Access Multiplexer - DSL) که بسیار شبیه مودم ADSL است، می‌رود. پس از تبدیل این سیگنال به جریان بیتهای دیجیتال، از آنجا به ISP فرستاده می‌شود.

جداسازی کامل سیستم صدا از ADSL ، پیاده‌سازی آنرا برای شرکتهای تلفن بسیار ساده کرده است: تمام کاری که باید انجام دهد اینست که در سمت خود یک تقسیم‌کننده و DSLAM ، و در سمت مشتری یک تقسیم‌کنندهٔ ساده نصب کند. در سیستمهای پُرسرعت دیگر (مانند، ISDN) تجهیزات بسیار پیچیده‌تری باید نصب شود.

یکی از معایب طرح شکل ۲-۲۹ وجود تجهیزاتی است که باید در محل سکونت مشترک نصب شود (NID و تقسیم‌کننده). این کار مستلزم آنست که یک تکنسین به آنجا مراجعه کرده، و وسایل را نصب کند. بهمین دلیل استاندارد جدیدی که به تقسیم‌کننده نیازی ندارد، توسعه داده شده است. نام غیررسمی این استاندارد G.lite است، ولی رسماً به آن ITU G.992.2 گفته می‌شود. این طرح شبیه شکل ۲-۲۹ است، که فقط تقسیم‌کننده حذف شده، و از خط تلفن بهمان صورت موجود استفاده می‌شود. تنها تفاوت اینست که باید میکروفیلترهایی را بین پریز تلفن و تجهیزات (دستگاه تلفن و مودم ADSL) قرار داد. میکروفیلتر تلفن یک فیلتر **پائین‌گذر** (low-pass) است که فرکانسهای بالای 3400 Hz را حذف می‌کند؛ میکروفیلتر مودم ADSL یک فیلتر **بالاگذر** (high-pass) است که فرکانسهای زیر 26 kHz را حذف می‌کند. با این حال کارایی G.lite مانند زمانی که تقسیم‌کننده بکار می‌رود، نیست و تنها می‌تواند به سرعت 1.5 MHz برسد (ولی هزاران مراجعهٔ مستقیم به منازل مشترکان را حذف می‌کند). در استاندارد G.lite به تقسیم‌کنندهٔ ایستگاه پایانی همچنان نیاز هست.

باید بیاد داشت که، ADSL یک استاندارد لایهٔ فیزیکی است، و آنچه که روی آن اجرا می‌شود به کاربر تلفن بستگی دارد. یکی از این کاربردها ATM است، چون در ATM امکان کنترل کیفیت وجود دارد و بسیاری از شرکتهای تلفن زیرساختهای آنرا در اختیار دارند.

مدار پایانی بیسیم

از سال ۱۹۹۶ در ایالات متحده آمریکا (و کمی بعد در کشورهای دیگر)، شرکتهایی که مایل بودند وارد رقابت با انحصارگر تلفن شهری (که به ILEC شهرت داشت) شوند، اجازهٔ آنرا یافتند. شرکتهای تلفن راه‌دور (IXC ها) اولین کسانی بودند، که وارد گود شدند. هر IXC که می‌خواست وارد بازار تلفن شهری شود، باید کارهای ذیل را انجام می‌داد. اول، زمین یا ساختمانی برای تأسیس ایستگاه پایانی (end office) بخرد یا اجاره کند. دوم، تجهیزات و سوئیچهای لازم را خریداری و در ایستگاه پایانی نصب کند. سوم، یک (یا چند) رشته فیبر نوری بین این ایستگاه پایانی و نزدیکترین ایستگاه بین شهری (tool office) بکشد، تا مشترکان آن بتوانند به شبکهٔ تلفن کشوری دسترسی داشته باشند. چهارم، در صدد جلب مشتری برآید (با تبلیغ و ارائهٔ سرویسهای بهتر و قیمت کمتر). و قسمت سخت کار همین جاست.

فرض کنید یک مشتری پیدا شده و می‌خواهد از شرکت تلفن شهری جدید (که آنرا CLEC می‌نامیم) سرویس تلفن بخرد. این شرکت چگونه باید مشتری را به ایستگاه پایانی خود - که خیلی هم برای آن خرج کرده - وصل کند؟ خریدن زمین در تمام طول مسیر، کندن کانال، و کشیدن یک خط تلفن بسیار پُرهزینه است. بسیاری از این CLEC ها راه ساده‌تر و کم‌هزینه‌تری پیدا کرده‌اند: WLL (**مدار پایانی بیسیم** - Wireless Local Loop).

از یک نظر تلفنهای ثابت با مدار پایانی بیسیم شبیه تلفنهای همراه هستند، ولی سه تفاوت فنی مهم و اساسی با آنها دارند: اول، مشترکان WLL دسترسی اینترنت پُرسرعت (با سرعتی حداقل معادل ADSL) می‌خواهند. دوم، این قبیل مشترکان انتظار ندارند (و اغلب اجازه نمی‌دهند) یک آنتن بشقابی بزرگ بالای بام خانه‌هایشان نصب شود. سوم، جای این تلفنها اساساً ثابت است، و مشکلات جابجا شدن سیگنال را (که در تلفنهای همراه وجود دارد، و بعداً در همین فصل با آنها آشنا خواهید دید) ندارند. بدین ترتیب یک صنعت کاملاً جدید متولد می‌شود: **بیسیم ثابت** (سرویس تلفن معمولی و اینترنت روی مدار پایانی بیسیم).

با اینکه آغاز بکار جدی WLL از سال ۱۹۹۸ است، ولی منشأ آن به سال ۱۹۶۹ برمی‌گردد. در این سال، FCC دو کانال تلویزیونی (هر یک با پهنای باند 6 MHz در فرکانس 2.1 GHz) به مصارف آموزشی اختصاص داد. در سالهای بعد تعداد این کانالها به ۳۳ (و پهنای باند مجموع آنها به 198 MHz در 2.5 GHz) افزایش یافت.

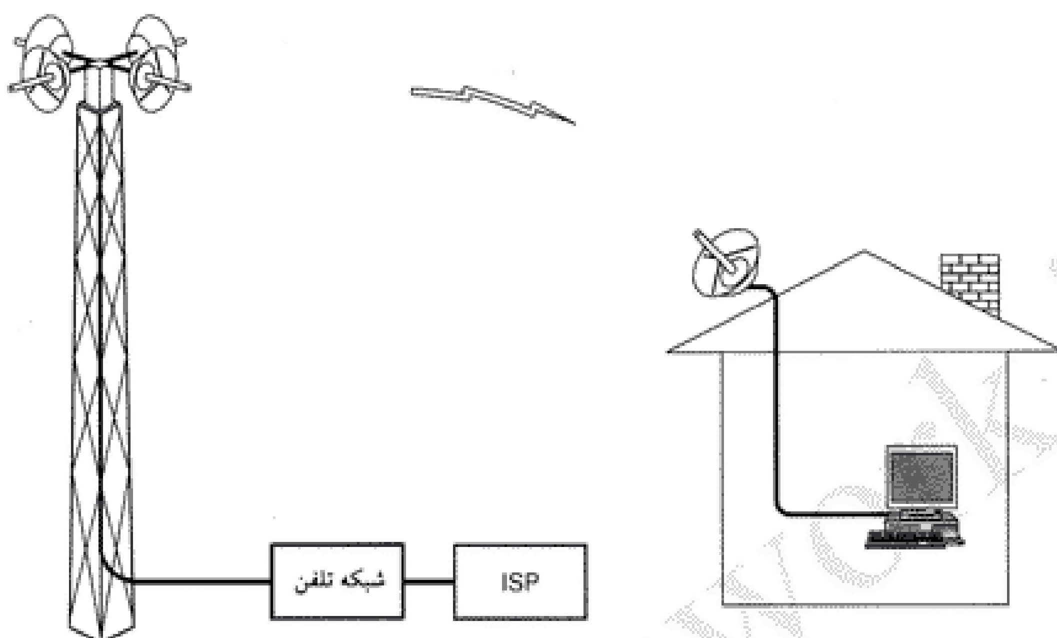
تلویزیون آموزشی هرگز پا نگرفت و در سال ۱۹۹۸، FCC این کانالها را پس گرفت و به ارتباطات رادیویی دوطرفه اختصاص داد - که بلافاصله در مدارهای پایانی بیسیم بکار گرفته شد. در این فرکانسها، طول موج امواج 10-12 cm است. بُرد این امواج حدود ۵۰ کیلومتر است، و از گیاهان و قطرات باران نسبتاً خوب عبور می‌کنند. سرویس جدیدی، که MMDS (**سرویس توزیع چندکاناله چندنقطه‌ای** - Multichannel Multipoint Distribution Service) نام گرفت، تمامی پهنای باند 198 MHz را مصرف کرد؛ MMDS را می‌توان (مانند پسر عمویش LMDS - که در همین قسمت آنرا توضیح می‌دهیم) یک شبکهٔ شهری (MAN) در نظر گرفت.

مزیت بزرگ این سرویس تکنولوژی جا افتاده و موجود بودن تجهیزات آن است؛ و بزرگترین عیب آن این است که پهنای باند متوسطی دارد، که آن هم باید بین افراد زیادی در یک محدودهٔ جغرافیایی وسیع به اشتراک گذاشته شود.

پهنای باند پائین MMDS باعث جذب شرکتهای تلفن به امواج میلیمتری شد. در فرکانسهای 28-31 GHz در آمریکا (و 40 GHz در اروپا) تمام باندها آزاد هستند، چون تکنولوژی تجهیزات نیمه‌هادی که در این فرکانسها کار کنند، پیچیده و گرانقیمت است. اما این مشکل هم با اختراع **مدارات مجتمع گالیوم-آرسناید** (Ga-As IC) حل شده، و راه برای ارتباطات رادیویی در باندهای میلیمتری هموار شده است. در پاسخ به تقاضاهای موجود، FCC یک پهنای باند 1.3 GHz به سرویس بیسیم جدید بنام LMDS (**سرویس توزیع چندنقطه‌ای محلی** - Local Multipoint Distribution Service) اختصاص داد - که این بزرگترین پهنای باندی بود که FCC تا به آن روز به یک کاربرد خاص اختصاص می‌داد. پهنای باند مشابهی نیز در اروپا (در فرکانس 40 GHz) به این سرویس اختصاص داده شده است.

عملکرد LMDS را در شکل ۲-۳۰ ملاحظه می‌کنید. در این شکل یک برج مخابراتی می‌بینید، که تعدادی آنتن در جهتهای مختلف روی آن نصب شده است. از آنجائیکه امواج میلیمتری بشدت خطی هستند، هر آنتن زاویهٔ محدودی (که به **قطاع** معروف است) را پوشش می‌دهد، که مستقل از سایر آنتنهاست. در این فرکانس، بُرد امواج 2-5 km است، و بهمین دلیل برای پوشش کامل یک شهر به تعداد زیادی از این برجهای مخابراتی نیاز هست.

در سرویس LMDS نیز، مانند ADSL، تخصیص پهنای باند نامتقارن (و به نفع کانال دریافت) است. با تکنولوژی موجود، هر قطاع می‌تواند تا 36 Mbps روی کانال دریافت کاربر و تا 1 Mbps روی کانال ارسال کاربر ظرفیت داشته باشد، که بین تمام کاربران موجود در آن قطاع به اشتراک گذاشته می‌شود. اگر هر کاربر در هر دقیقه سه صفحهٔ 5-KB دریافت کند، بطور متوسط 2000 bps از پهنای باند را اشغال می‌کند، که بدین ترتیب همزمان حداکثر 18,000 کاربر می‌توانند در هر قطاع کار کنند. البته برای قابل قبول بودن سرویس، نباید در هر لحظه بیش از ۹۰۰۰ کاربر فعال داشته باشیم. با فرض داشتن چهار قطاع در یک برج (مانند شکل ۲-۳۰)، تعداد کاربرانی که در



شکل ۲-۳۰. معماری یک سیستم LMDS .

هر لحظه می‌توانند فعال باشند به ۳۶,۰۰۰ می‌رسد - و اگر فرض کنیم در ساعات اوج مصرف از هر سه کاربر یکی فعال است، تعداد کل مشترکانی که یک برج مخابراتی در شعاع ۵کیلومتری می‌تواند پوشش دهد، به ۱۰۰,۰۰۰ بالغ می‌شود. (بر مبنای همین محاسبات بود که تعداد زیادی از CLEC ها به این نتیجه رسیدند که با یک سرمایه‌گذاری متوسط در برجهای مخابراتی امواج میلیمتری، می‌توانند در بازار تلفن شهری و اینترنت با شرکتهای تلویزیون کابلی رقابت کنند.)

اما، LMDS هم مشکلات خاص خود را دارد. اول اینکه، امواج میلیمتری کاملاً خطی هستند، و هیچ مانعی در خط دید برج مخابراتی و آنتن گیرنده نبایستی وجود داشته باشد. دیگر اینکه، برگ درختان جاذب خوبی برای این امواج است، بنابراین برج فرستنده بایستی آنقدر مرتفع باشد که درختان مانع امواج آن نشوند (و باید توجه داشت مسیری که در زمستان صاف است، ممکنست در تابستان که درختان پر از برگ هستند، چنین نباشد). باران نیز امواج میلیمتری را جذب می‌کند. این مشکل را می‌توان تا حدی با کُدهای تصحیح خطا، و افزایش توان تشعشعی فرستنده در روزهای بارانی جبران کرد. معهذا، سرویس LMDS در مناطق خشک شانس بیشتری برای موفقیت دارد، تا مناطق مرطوب و پُر باران.

مدار پایانی بیسیم بدون وجود استانداردهای معتبر شانسی برای موفقیت ندارد، چون فقط در صورت وجود این استانداردهاست که سازندگان تجهیزات الکترونیکی رغبت ساخت وسایل مورد نیاز این صنعت را پیدا می‌کنند، و مشترکان نیز می‌توانند با خیال راحت (بدون نگرانی از اینکه با عوض کردن شرکت تلفن، وسایل را هم باید تعویض کنند) مشترک این سرویسها شوند. بمنظور تدوین استانداردهای LMDS ، کمیتهٔ 802.16 توسط IEEE تأسیس شد، که این استاندارد را (که **شبکهٔ شهری بیسیم** نامیده می‌شود) در آوریل ۲۰۰۲ عرضه کرد.

استاندارد IEEE 802.16 تلفن دیجیتال، دسترسی اینترنت، ارتباط بین شبکه‌های محلی، ایستگاههای پخش برنامه‌های رادیویی و تلویزیونی (و چندین کاربرد دیگر) را در بر می‌گیرد. در فصل ۴ دربارهٔ این استاندارد بیشتر صحبت خواهیم کرد.

۲-۵-۴ ترانک‌ها و مالتی‌پلکس کردن

ساخت و ساز نقش مهمی در اقتصاد سیستمهای تلفن بازی می‌کند. نصب یک **خط اصلی با پهنای باند زیاد** (high-bandwidth trunk) بین دو مرکز سوئیچینگ به همان اندازهٔ یک **خط اصلی با پهنای باند کم** (low-bandwidth trunk) پول لازم دارد، چون هزینهٔ اصلی در اینجا هزینهٔ حفر کانالهاست نه هزینهٔ کابل مسی یا فیبر نوری. بهمین دلیل، شرکتهای تلفن از روشهای پیچیده‌ای برای ارسال همزمان چندین مکالمه روی یک خط فیزیکی استفاده می‌کنند، که به این روشها **مالتی‌پلکس** (multiplex) گفته می‌شود. تکنیکهای مالتی‌پلکس کردن به دستهٔ زرگ تقسیم می‌شوند: FDM (**مالتی‌پلکس تقسیم فرکانس** – Frequency Division Multiplexing) و TDM (**مالتی‌پلکس تقسیم زمان** – Time Division Multiplexing). در FDM ، طیف فرکانسی به باندهای مختلفی تقسیم می‌شود، و هر کاربر انحصاراً از یک باند استفاده می‌کند. در TDM، هر کاربر برای لحظه‌ای کوتاه (برش کوچکی از زمان) کل پهنای باند را در اختیار می‌گیرد.

در رادیوهای AM هر دو نوع مالتی‌پلکس دیده می‌شود. طیف تخصیص داده شده به هر کانال (ایستگاه) در حدود 1 MHz (تقریباً بین 500-1500 kHz) است، و هر ایستگاه در باند اختصاصی خود کار می‌کند. بین ایستگاهها نیز آنقدر فاصله وجود دارد، که با هم تداخل نکنند. این سیستم نمونه‌ای از FDM است. علاوه بر آن، در برخی از کشورها هر ایستگاه دارای دو زیرکانال منطقی است: کانال موزیک و کانال آگهی. فرستنده در برشهای کوتاه زمانی (و بصورت یک در میان) موزیک و آگهی پخش می‌کند، که در این حالت بصورت TDM کار می‌کند.

در ادامه، ابتدا مالتی‌پلکس تقسیم فرکانس (و کاربرد آن در فیبرهای نوری، که به مالتی‌پلکس تقسیم طول موج معروفست) را خواهید دید. سپس با مالتی‌پلکس تقسیم زمان (و یکی از کاربردهای پیشرفتهٔ آن در فیبرهای نوری، بنام SONET) آشنا می‌شوید.

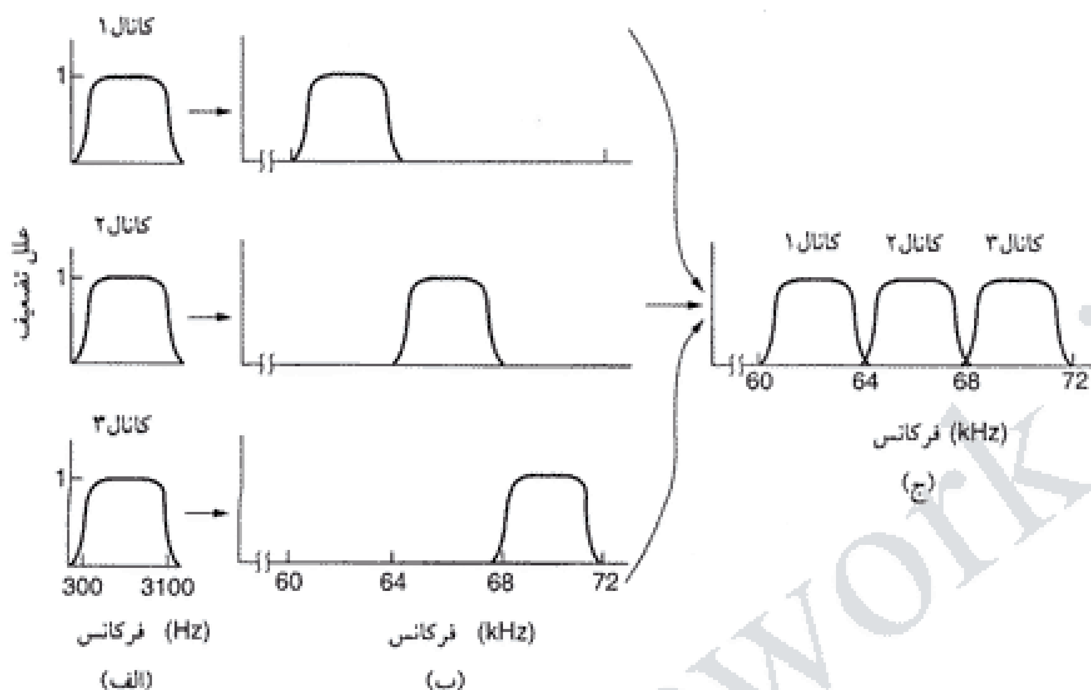
مالتی‌پلکس تقسیم فرکانس

در شکل ۲-۳۱ مالتی‌پلکس ۳ کانال صوتی با استفاده از FDM نشان داده شده است. پهنای باند هر کانال با استفاده از فیلترهای مخصوص به 3100 Hz محدود شده، ولی هنگام مالتی‌پلکس کردن پهنای هر کانال 4000 Hz در نظر گرفته می‌شود تا بین آنها تداخل پیش نیاید. در اولین قدم، فرکانس هر کانال به مقدار مشخصی (که با سایر کانالها متفاوت است) بالا برده می‌شود. سپس می‌توان این کانالها را با هم ترکیب کرد، چون دیگر هیچکدام از آنها فرکانس یکسانی ندارند و با هم مخلوط نمی‌شوند. دقت کنید که با وجود در نظر گرفتن یک حاشیهٔ امنیتی برای هر کانال، آنها تا حدی روی هم می‌افتند، چون نقطهٔ قطع فیلترها کاملاً تیز نیست. این روی هم افتادگی باعث بروز نوعی نویز غیرحرارتی در کانالهای مجاور می‌شود.

روشهای FDM که در سرتاسر دنیا بکار برده می‌شوند، تا حدی استاندارد شده‌اند. یکی از این استانداردها مالتی‌پلکس کردن دوازده کانال صوتی 4000-Hz روی باند 60-108 kHz است. این واحد **گروه** (group) نامیده می‌شود. گاهی اوقات از باند 12-60 kHz نیز بعنوان یک گروه استفاده می‌شود. بسیاری از شرکتهای تلفن سرویسهای خطوط اجاره‌ای 48 kbps تا 56 kbps را بصورت همین گروهها به مشتریان خود ارائه می‌کنند. از مالتی‌پلکس پنج گروه (یعنی ۶۰ کانال صوتی) یک **فوق‌گروه** (supergroup) بوجود می‌آید؛ و از مالتی‌پلکس پنج (در استاندارد CCITT) یا ده (در استاندارد شرکت بل) فوق‌گروه یک **ابَرگروه** (mastergroup) شکل می‌گیرد. در برخی از استانداردها حتی تا ۲۳۰,۰۰۰ کانال صوتی در یک باند مالتی‌پلکس می‌شود.

مالتی‌پلکس تقسیم طول‌موج

برای کانالهای فیبر نوری از نوع دیگری از مالتی‌پلکس تقسیم فرکانس استفاده می‌شود، که **مالتی‌پلکس تقسیم**



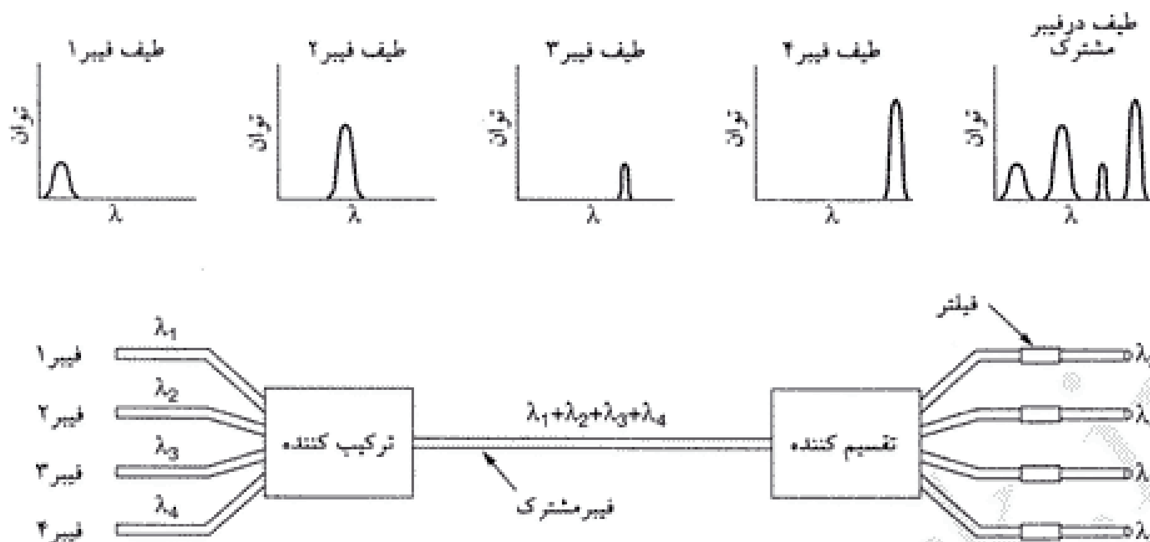
شکل ۲-۳۱. مالتی‌پلکس تقسیم فرکانس. (الف) کانالهای اولیه. (ب) بالا بردن فرکانس کانالها. (ج) کانالهای مالتی‌پلکس شده.

طول‌موج (Wavelength Division Multiplexing) نام دارد. اصول WDM در شکل ۲-۳۲ نشان داده شده است. در اینجا پرتوهای چهار فیبر نوری، که هر کدام طول‌موج متفاوتی دارند، وارد یک ترکیب‌کنندهٔ نوری شده، و برای ارسال آماده می‌شوند. در انتهای دیگر نیز عمل عکس انجام شده، و پرتوها مجدداً تفکیک می‌شوند. این کار توسط فیلترهای مخصوصی که فقط به یک طول‌موج اجازه عبور می‌دهند، انجام می‌شود.

در واقع هیچ چیز جدیدی در این تکنیک وجود ندارد، و این یک مالتی‌پلکس تقسیم فرکانس در فرکانسهای بسیار بالاست. مادامیکه هر کانال دارای فرکانس (یا، طول‌موج) خاص خود باشد، می‌توان آنها را با هم ترکیب (مالتی‌پلکس) کرد. تنها فرق این روش با FDM الکتریکی اینست که در سیستمهای نوری از فیلترهای انکساری منفعل استفاده می‌شود، که بسیار قابل‌اطمینان هستند.

سرعت پیشرفت تکنولوژی WDM آنقدر زیاد است که صنعت کامپیوتر به گرد آن هم نمی‌رسد. این تکنولوژی در سال ۱۹۹۰ اختراع شد، و اولین سیستم تجاری WDM دارای هشت کانال 2.5 Gbps بود. در سال ۱۹۹۸، سیستمهایی با ۴۰ کانال 2.5 Gbps به بازار عرضه شد، و در سال ۲۰۰۱ شاهد سیستمهایی با ۹۶ کانال 10 Gbps بودیم، که ظرفیت کل آنها به 960 Gbps می‌رسد (این پهنای باند برای ارسال ۳۰ فیلم کامل - با فرمت MPEG-2 - در هر ثانیه کافیست). در آزمایشگاهها سیستمهایی با ۲۰۰ کانال نیز تست شده‌اند. وقتی تعداد کانالها خیلی زیاد باشد، و طول‌موجها فاصلهٔ کمی (در حد 0.1 nm) با هم داشته باشند، به آن DWDM (WDM **چگال** - Dense WDM) گفته می‌گویند.

علت محبوبیت سیستمهای WDM اینست که انرژی هر فیبر نوری فقط چند گیگاهرتز پهنا دارد، چون در حال حاضر امکان تبدیل سریعتر سیگنالهای الکتریکی به پالسهای نوری (و بالعکس) وجود ندارد. با ترکیب چند کانال با طول‌موجهای مختلف، پهنای باند بصورت خطی افزایش پیدا می‌کند. از آنجائیکه پهنای باند یک فیبر نوری در حدود 25,000 GHz است (شکل ۲-۶ را ببینید)، ظرفیت آن می‌تواند به ۲۵۰۰ کانال 10-GHz برسد - و این



شکل ۲-۳۲. مالتی‌پلکس تقسیم طول‌موج.

در bit/Hz1 است، که البته نرخهای بالاتر هم امکان دارد.

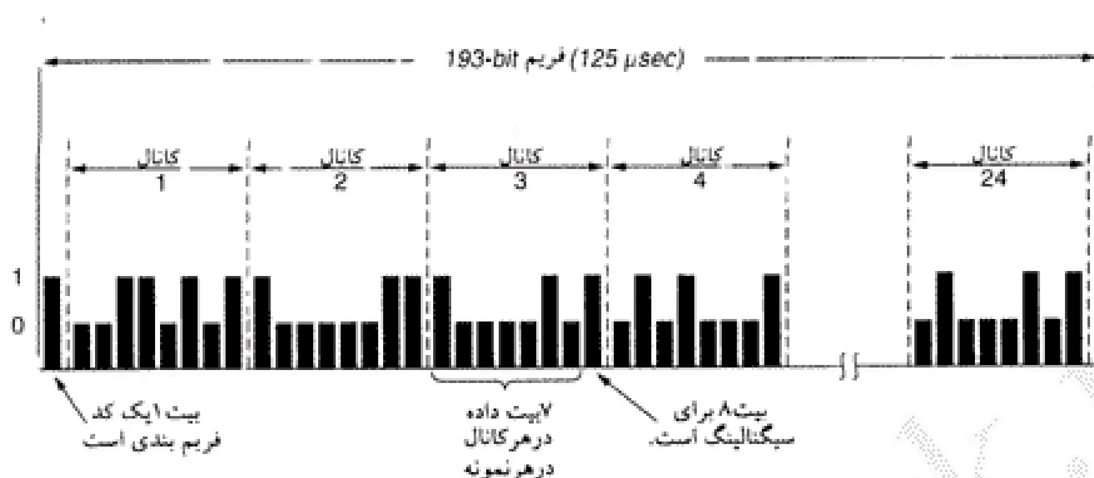
پیشرفت جدیدی که در این زمینه بدست آمده، تقویت‌کننده‌های تمام نوری است. در گذشته، لازم بود تا در فواصل ۱۰۰ کیلومتری پالسهای نوری به سیگنالهای الکتریکی تبدیل شده، و پس از تقویت دوباره به پالسهای نوری تبدیل و ارسال شوند. امروزه، تقویت‌کننده‌های تمام نوری می‌توانند (بدون نیاز به تبدیل‌کننده‌های نوری-الکتریکی) عمل تقویت را در هر ۱۰۰۰ کیلومتر انجام دهند.

مثال شکل ۲-۳۲ یک سیستم ثابت است: بیتهای فیبر ۱ به خروجی ۳ می‌روند، بیتهای فیبر ۲ به خروجی ۱ می‌روند، و الی‌آخر. ولی، ساخت سیستمهای سوئیچینگ WDM نیز امکانپذیر است. در این قبیل دستگاهها، می‌توان فیلترهای خروجی را با استفاده از تداخل‌سنجهای **فابری-پروت** یا **ماخ-زندر** تنظیم کرد. برای کسب اطلاعات بیشتر در زمینهٔ WDM و کاربرد آن در سوئیچنگ بستهٔ اینترنت، به (Elmirghani and Mouftah, 2000; Hunter and Andonovic, 2000; Listani et al., 2001) مراجعه کنید.

مالتی‌پلکس تقسیم زمان

تکنولوژی WDM بسیار جالب و مهیج است، ولی هنوز هزاران هزار کیلومتر سیم مسی در شبکهٔ تلفن موجود است که باید فکری هم بحال آنها کرد. با اینکه FDM هنوز کاربرد گسترده‌ای در کابلهای مسی و کانالهای مایکروویو دارد، اما این تکنولوژی اساساً آنالوگ است و نمی‌توان آنرا با کامپیوتر انجام داد. ولی TDM (**مالتی‌پلکس تقسیم زمانی** – Time Division Multiplexing) را می‌توان بطور کامل دیجیتالی کرد، و بهمین دلیل در سالهای اخیر کاربرد گسترده‌ای یافته است. متاسفانه، از TDM فقط برای داده‌های دیجیتال می‌توان استفاده کرد، و از آنجائیکه مدارهای پایانی سیگنالهای آنالوگ تولید می‌کنند، باید این سیگنالها را در ایستگاه پایانی به دیجیتال تبدیل کرده، و سپس روی ترانک‌ها (که دیجیتال هستند) ارسال کرد.

در این قسمت نحوهٔ دیجیتالی کردن سیگنالهای آنالوگ صدا، و ترکیب آنها برای ارسال روی ترانک‌های دیجیتال را توضیح می‌دهیم – سیگنالهایی که کامپیوترها از طریق مودم ارسال می‌کنند نیز آنالوگ است، بنابراین توضیحات زیر در مورد آنها هم صادق است. سیگنالهای آنالوگ در ایستگاه پایانی توسط دستگاهی بنام **کُدِک** (codec) دیجیتایز شده، و یکسری اعداد ۸بیتی تولید می‌شود. این کُدک در هر ثانیه ۸۰۰۰ نمونه می‌گیرد (در هر



شکل ۲-۳۳. کاربر T1 (1.544 Mbps).

۱۲۵ میکروثانیه یک نمونه). چون طبق قضیهٔ نایکوئیست این مقدار برای گرفتن تمام اطلاعات کانالی با پهنای باند 4-kHz کافیست - با نرخ نمونه‌برداری پائینتر اطلاعات از دست می‌رود، و با نرخ بالاتر اطلاعات بیشتری بدست نمی‌آید. این تکنیک که PCM (**مدولاسیون کُد پالس** - Pulse Code Modulation) نامیده می‌شود، قلب سیستمهای جدید تلفن است. در نتیجه، تمام فواصل زمانی در سیستمهای تلفن مضاربی از 125 µsec هستند. با ورود تکنولوژی مخابرات دیجیتال، CCITT نتوانست بر سر استانداردی بین‌المللی برای آن به توافق برسد، بهمین دلیل سیستمهای دیجیتال متعددی در سراسر دنیا مورد استفاده قرار گرفتند که عمدتاً با هم ناسازگار بودند.

تکنیکی که در آمریکای شمالی و ژاپن از آن استفاده شد، کاریر T1 است که آنرا در شکل ۲-۳۳ ملاحظه می‌کنید. (بخواهیم دقیقتر صحبت کرده باشیم، این فرمت DS1 و کاریر آن T1 خوانده می‌شوند، ولی در اینجا قصد نداریم با آنچه در صنعت جا افتاده مخالفت کنیم) هر کاریر T1 عبارتست از ۲۴ کانال صوتی که با یکدیگر مالتی‌پلکس شده‌اند. معمولاً، این سیگنالهای آنالوگ در فواصل زمانی منظم و متوالی به یک کُدک داده شده و دیجیتایز می‌شوند (بجای آنکه از ۲۴ کُدک استفاده کرده، و خروجی آنها را ترکیب کنیم). هر یک از این ۲۴ کانال بنوبت ۸ بیت به خروجی می‌دهند. با احتساب ۷ بیت داده و یک بیت کنترل، در هر کانال $7 \times 8000 = 56{,}000$ bps داده و $1 \times 8000 = 8000$ bps اطلاعات کنترلی خواهیم داشت.

هر فریم دارای $24 \times 8 = 192$ بیت است، که (با احتساب یک بیت اضافی برای فریم‌بندی) 193 بیت در هر 125 µsec خواهیم داشت، که بدین‌ترتیب نرخ داده 1.544 Mbps به می‌رسد. بیت ۱۹۳ ام برای سنکرون کردن فریم مورد استفاده قرار می‌گیرد. این بیت شکل ...0101010101 بخود می‌گیرد، و گیرنده بکمک این بیت می‌تواند از سنکرون بودن اطلاعات اطمینان یابد. اگر گیرنده همزمانی خود را با فرستنده از دست دهد، می‌تواند با جستجوی این طرح بیت دوباره با فرستنده سنکرون شود. کاربران آنالوگ بکلی نمی‌توانند چنین طرحی از بیت‌ها تولید کنند، چون این طرح بیت معادل موج سینوسی 4000-Hz است که حذف خواهد شد. کاربران دیجیتال می‌توانند چنین طرحی را تولید کنند، ولی مشکل اینجاست که این طرح با لغزش فریم نیز می‌تواند ظاهر شود. برای بازیابی سریعتر سیستم در صورت بروز چنین لغزش‌هایی، هنگامی که از T1 فقط برای ارسال داده استفاده می‌شود، کانال ۲۴ ام به یک طرح خاص سنکرون کردن اختصاص می‌یابد (و فقط در ۲۳ کانال داده ارسال می‌شود).

وقتی بالاخره CCITT بر سر استانداردهای PCM بتوافق دست یافت، احساس کرد که ۸۰۰۰ بیت کنترلی

خیلی زیاد است، بنابراین استاندارد 1.544-Mbps بر اساس بسته‌های دادهٔ ۸ بیتی بنا شد نه ۷ بیتی؛ بعبارت دیگر، هر سیگنال آنالوگ بجای ۱۲۸ سطح به ۲۵۶ سطح مجزا دیجیتایز می‌شود. دو ویرایش (ناسازگار) از این استاندارد تهیه شد. در **سیگنالینگ کانال مشترک** (common-channel signaling) بیت اضافی (که در اینجا بجای ابتدا به انتهای فریم ۱۹۳ بیتی چسبانده می‌شود) در فریمهای فرد مقدار ...10101010 بخود می‌گیرد، و در فریمهای زوج (تمام کانالها) حاوی اطلاعات سیگنالینگ است.

در ویرایش دیگر، **سیگنالینگ وابسته به کانال** (channel-associated signaling)، هر کانال دارای زیرکانال سیگنالینگ مخصوص بخود است. در هر زیرکانال یک بیت از هر هشت بیت داده در هر شش فریم به سیگنالینگ اختصاص داده می‌شود، بنابراین از هر شش فریم پنج تای آنها ۸ بیتی و یکی ۷ بیتی است. یکی دیگر از پیشنهادات CCITT، کاریر PCM با ظرفیت 2.048 Mbps است که E1 نام دارد. این کاریر دارای ۳۲ بستهٔ ۸ بیتی در هر 125 μsec است، که ۳۰ تای آنها برای داده و دو تا برای سیگنالینگ بکار می‌روند. در هر گروه چهار فریمی ۶۴ بیت سیگنالینگ وجود دارد، که نیمی از آن به سیگنالینگ وابسته به کانال اختصاص داده شده، و نیمی دیگر برای سنکرون کردن فریمها کنار گذاشته شده است (کشورهای مختلف می‌توانند از این نیمه بدلخواه استفاده کنند). خارج از آمریکای شمالی و ژاپن بجای T1 از کاریر E1 استفاده می‌شود.

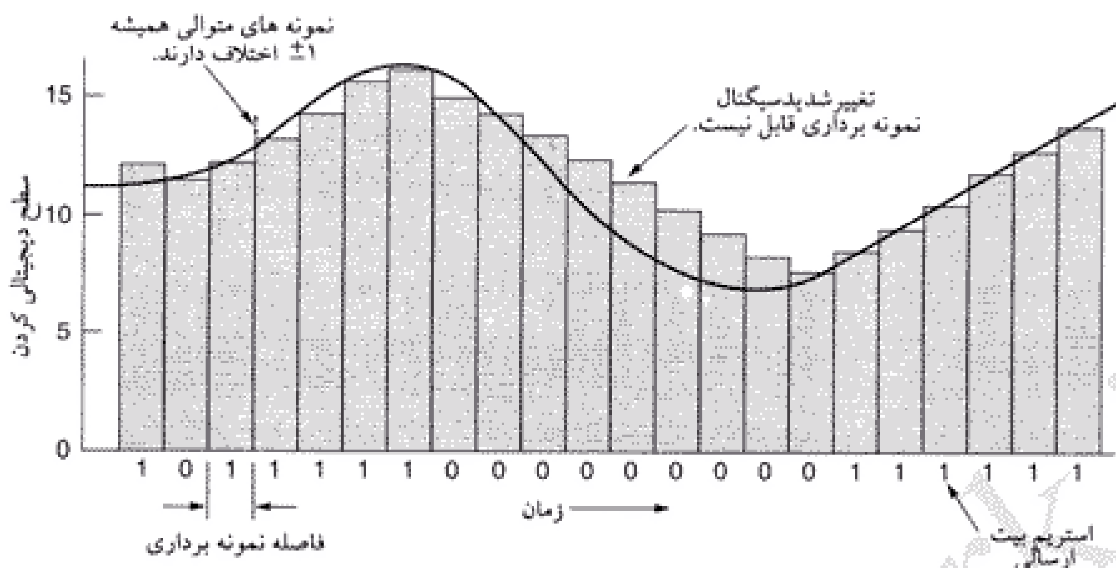
بعد از دیجیتایز کردن سیگنال صوتی، می‌توان با استفاده از تکنیکهای آماری تعداد بیتهای مورد نیاز هر کانال را کاهش داد (فشرده‌سازی). از این تکنیکها برای دیجیتایز کردن هر نوع سیگنال آنالوگ می‌توان استفاده کرد. تمام تکنیکهای فشرده‌سازی بر این اصل استوارند که تغییرات سیگنال نسبت به فرکانس نمونه‌برداری نسبتاً کمتر است، بنابراین قسمت اعظم سطوح دیجیتال ۷ یا ۸ بیتی تکراری و زائد هستند.

در یکی از این روشها، که **مدولاسیون کُد پالس تفاضلی** (differential pulse code modulation) نام دارد، نه خود دامنهٔ دیجیتایز شده بلکه تفاوت آن با مقدار قبلی بکار برده می‌شود. از آنجائیکه در یک مقیاس ۱۲۸ تایی پرشهایی بیشتر از 16 ± ممکن نیست، بجای ۷ بیت به فقط ۵ بیت نیاز داریم. اگر سیگنال در مواردی خاص بیش از ۱۶ سطح اختلاف داشته باشد، برای نمونه‌برداری آن به بیش از یک دورهٔ زمانی نیاز هست. خطایی که بدین‌ترتیب روی می‌دهد، در دیجیتایز کردن صدا قابل چشم‌پوشی است.

در نوع اصلاح شدهٔ این تکنیک فشرده‌سازی، هر نمونه باید با نمونهٔ قبلی 1 ± اختلاف داشته باشد. در این شرایط بیتی که فرستاده می‌شود برای آنست که مشخص کند، نمونهٔ جدید بیشتر از قبلی است یا کمتر از آن. این تکنیک را، که **مدولاسیون دلتا** (delta modulation) نامیده می‌شود، در شکل ۲-۳۴ ملاحظه می‌کنید. مانند سایر تکنیکهای فشرده‌سازی (که فرض را بر تغییرات اندک در سیگنال آنالوگ می‌گذارند) مدولاسیون دلتا هم در تغییرات شدید سیگنال دچار خطا می‌شود (که در چنین مواردی اطلاعات از دست می‌رود).

روش بهبود یافتهٔ PCM تفاضلی بر پیش‌بینی مقدار بعدی سیگنال با استفاده از برون‌یابی چند مقدار قبلی، و سپس محاسبهٔ تفاضل این مقدار پیش‌بینی شده با مقدار واقعی سیگنال استوار است - به این روش **کُد کردن پیشگویانه** (predictive encoding) گفته می‌شود. البته در این روش فرستنده و گیرنده هر دو باید از روش پیش‌بینی یکسانی استفاده کنند. روش کُد کردن پیشگویانه باعث کم شدن اندازهٔ اعدادی که باید کُد شوند، می‌شود و بهمین دلیل تعداد بیتهای ارسالی کاهش خواهد یافت.

تکنیک TDM اجازه می‌دهد تا چندین کاریر T1 روی یک کاریر مرتبهٔ بالاتر مالتی‌پلکس شوند - در شکل ۲-۳۵ این تکنیک را ملاحظه می‌کنید. در اینجا چهار کانال T1 (سمت چپ) روی یک کانال T2 مالتی‌پلکس شده‌اند. در کانال T1 مالتی‌پلکس بصورت بایت به بایت انجام می‌شود، ولی از T2 به بعد مالتی‌پلکس بیت به بیت صورت می‌گیرد. چهار استریم T1 که هر کدام 1.544 Mbps هستند، بایستی خروجی 6.167 Mbps تولید کند،



شکل ۲-۳۴. مدولاسیون دلتا.

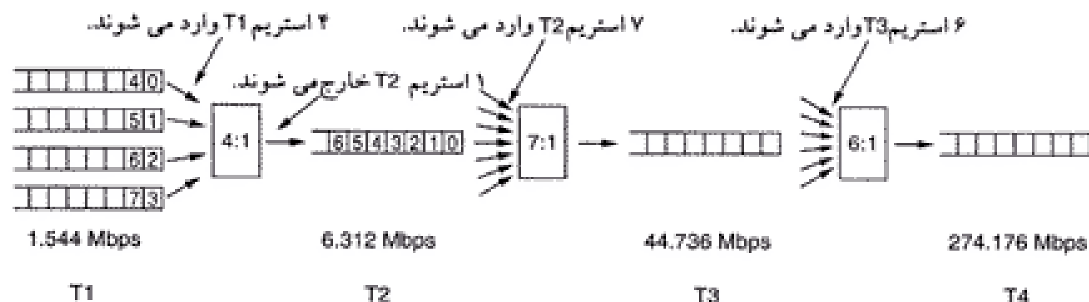
ولی ظرفیت T2 در واقع 6.312 Mbps است. بیتهای اضافی برای فریم‌بندی و بازیابی کانال در صورت لغزش کاریر منظور شده‌اند. کاریرهای T1 و T3 در میان مشترکین شناخته‌شده‌تر هستند، چون T2 و T4 بیشتر برای مصارف داخلی سیستم تلفن بکار می‌رود.

در مرحلهٔ بعد، هفت استریم T2 بصورت بیت به بیت روی یک استریم T3 مالتی‌پلکس می‌شوند؛ از ترکیب شش استریم T3 هم یک T4 بوجود می‌آید. در هر مرحله مقدار کمی سرآیند نیز برای فریم‌بندی و تصحیح خطای لغزش کاریر اضافه می‌شود.

همانطور که بر سر کاریر اصلی توافق کمی بین ایالات متحده و سایر کشورهای جهان وجود دارد، نحوهٔ مالتی‌پلکس آنها نیز استاندارد دقیقی ندارد. کمتر کسی را در دنیا پیدا می‌کنید که با ضرایب پیشنهادی آمریکا (۴، ۷ و ۶) موافق باشد، بهمین دلیل CCITT از مضرب یکنواخت ۴ در تمام مراحل استفاده می‌کند. همچنین نحوهٔ فریم‌بندی و بازیابی کانال هم در ایالات متحده و CCITT متفاوت است. در استاندارد پیشنهادی CCITT تعداد کانالها بترتیب ۳۲، ۱۲۸، ۵۱۲، ۲۰۴۸ و ۸۱۹۲ است، که با سرعتهای 2.048 Mbps، 8.848 Mbps، 34.304 Mbps، 139.264 Mbps، و 565.148 Mbps کار می‌کنند.

SONET/SDH

در روزهای اولیهٔ فیبر نوری، هر شرکت تلفن برای خود یک سیستم TDM نوری اختصاصی داشت. بعد از تجزیهٔ



شکل ۲-۳۵. مالتی‌پلکس استریم T1 روی کاریرهای مرتبهٔ بالاتر.

AT&T در سال ۱۹۸۴، شرکتهای تلفن شهری مجبور شدند به کاریرهای بین‌شهری مختلف که هر کدام سیستم TDM نوری خاص خود را داشت متصل شوند، بهمین دلیل به فکر استاندارد کردن آن افتادند. در ۱۹۸۵، شرکت Bellcore (بازوی تحقیقاتی شرکتهای RBOC) کار روی استانداردی بنام SONET (**شبکهٔ نوری سنکرون** - Synchronous Optical NETwork) را شروع کرد. بعدها CCITT نیز وارد این معرکه شد، که نتیجهٔ آن استانداردهای موازی SONET ، G.707 و G.709 بود. توصیه‌های CCITT که SDH (**سلسله‌مراتب دیجیتال‌سنکرون** - Synchronous Digital Hierarchy) خوانده می‌شوند، فقط چند تفاوت جزیی با SONET دارند. امروزه تقریباً تمامی ترافیک تلفن راه دور در ایالات متحده و اکثر نقاط دنیا، از SONET در لایهٔ فیزیکی ترانک استفاده می‌کنند. برای کسب اطلاعات بیشتر دربارهٔ SONET به (Bellamy, 2000; Goralski, 2000; Shepard, 2001) مراجعه کنید.

در طراحی SONET چهار هدف اصلی مد نظر بوده است. اول و از همه مهمتر، SONET بایستی کاری می‌کرد که کاریرهای مختلف بتوانند با هم کار کنند. برای رسیدن به این هدف به یک استاندارد سیگنالینگ (در زمینه‌های طول‌موج، تایمینگ، ساختار فریم‌بندی و غیره) نیاز بود.

دوم، SONET باید راهی برای یکپارچه کردن سیستمهای دیجیتال آمریکایی، اروپایی و ژاپنی (که همگی از کانالهای 64-kbps PCM ، ولی با روشهای مختلف و ناسازگار، استفاده می‌کردند) پیدا می‌کرد.

سوم، SONET باید راهی برای مالتی‌پلکس کردن کانالهای دیجیتال متعدد پیدا می‌کرد. در زمان تدوین SONET سریعترین کاریر دیجیتال، که کاربرد گسترده‌ای در ایالات متحده داشت، T3 با سرعت 44.736 Mbps بود. کاریر T4 تعریف شده، ولی هنوز بطور کامل عملیاتی نشده بود (برای بالاتر از آن هم هنوز کسی فکری نکرده بود). بخشی از مأموریت SONET ادامهٔ راه تا gigabits/sec و بالاتر بود. همچنین باید راهی برای مالتی‌پلکس کردن کانالهای کُندتر در یک کانال SONET پیش‌بینی می‌شد.

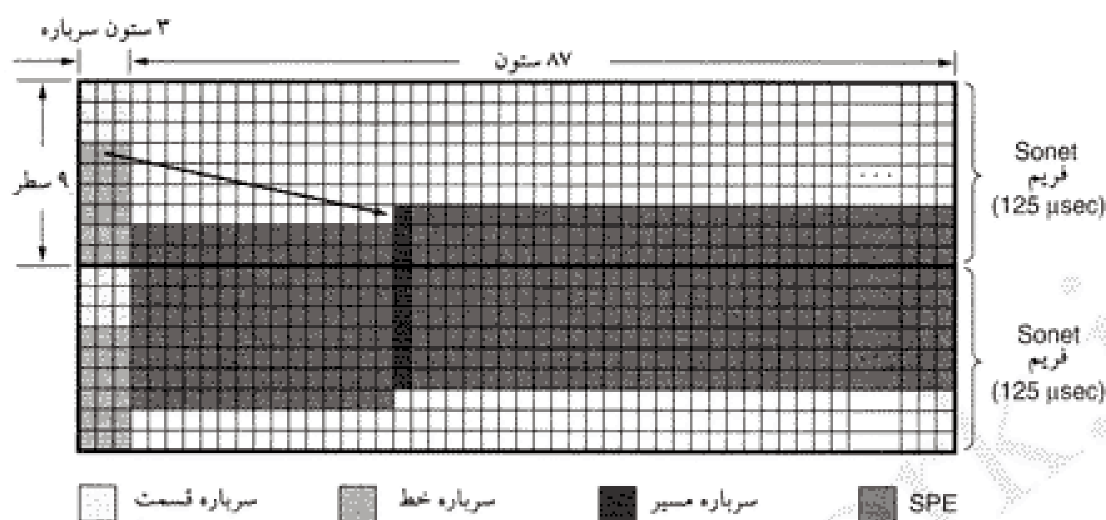
چهارم، SONET باید از یک سیستم جامع عملیات، مدیریت و نگهداری (OAM) پشتیبانی می‌کرد - وظیفه‌ای که در سیستمهای قبلی توجه چندانی به آن نشده بود.

از همان ابتدا تصمیم بر آن شد تا SONET یک سیستم سادهٔ TDM باشد، بگونه‌ایکه تمام پهنای باند فیبر نوری یک کانال واحد را تشکیل دهد، و هر زیرکانال برای کسری از ثانیه کانال را در اختیار بگیرد. بدین‌ترتیب، SONET یک سیستم سنکرون خواهد بود، که توسط یک ساعت اصلی (با دقت 1 در 10^9) کنترل می‌شود. در یک خط SONET بیت‌ها با فواصل زمانی فوق‌العاده دقیق (که توسط ساعت اصلی کنترل می‌شود) ارسال می‌شوند. وقتی بعدها سوئیچینگ سلول بعنوان مبنای ATM انتخاب شد، برای تمایز آن با روش **سنکرون** SONET ، از کلمهٔ آسنکرون (*Asynchronous* Transfer Mode) استفاده شد. در SONET ، فرستنده و گیرنده به یک ساعت مشترک وابسته‌اند، در حالیکه در ATM چنین نیست.

در SONET فریمها 810 بایتی هستند، که در فواصل زمانی 125 μsec ارسال می‌شوند. از آنجائیکه SONET یک سیستم سنکرون است، خواه داده‌ای باشد یا نباشد، فریمها روی خط فرستاده می‌شوند. با این نرخ (8000 frames/sec) ، SONET کاملاً با کانالهای PCM سازگار خواهد بود.

فریمهای ۸۱۰بایتی SONET را می‌توان جدولی با ۹۰ستون و ۹ سطر فرض کرد. بنابراین در هر ثانیه ۸۰۰۰ فریم ۶۴۸۰ = ۸۱۰ × ۸بیتی ارسال می‌شود، که سرعت کل آنرا به 51.84 Mbps می‌رساند. این کانال پایهٔ SONET است، که STS-1 (**سیگنال انتقال سنکرون ۱** - Synchronous Transport Signal-1) نامیده می‌شود. تمام ترانک‌های SONET مضاربی از STS-1 هستند.

همانطور که در شکل ۲-۳۶می‌بینید، سه ستون اول هر فریم به اطلاعات مدیریت سیستم اختصاص یافته‌اند.



شکل ۲-۳۶. دو فریم متوالی SONET .

در سه سطر اول این بایتها محتوی **سرآیند قسمت** (section overhead) هستند، و در شش سطر بعدی محتوی **سرآیند خط** (line overhead) . سرآیند قسمت در ابتدا و انتهای هر قسمت ایجاد و چک می‌شود، و سرآیند خط در ابتدا و انتهای خط.

فرستندهٔ SONET فریمهای ۸۱۰ بایتی را بصورت متوالی (back-to-back) ارسال می‌کند، حتی اگر هیچ داده‌ای در کار نباشد - که در این حالت اطلاعات ساختگی ارسال می‌شود. از دید گیرنده این یک استریم پیوسته ازبیتهاست، پس چگونه می‌تواند تشخیص دهد ابتدای هر فریم کجاست؟ پاسخ اینست که دو بایت ابتدایی هر فریم طرح خاصی دارند، که گیرنده می‌تواند آنرا تشخیص دهد. بمحض یافتن این طرح، گیرنده با فرستنده سنکرون می‌شود. شاید بگوئید که احتمال دارد این طرح بیت‌ها در داده‌های کاربر نیز وجود داشته باشد، اما از آنجائیکه داده‌های چندین کاربر در یک فریم مالتی‌پلکس می‌شوند (و به دلایل دیگر) این اتفاق نخواهد افتاد.

ستونهای باقیمانده (۸۷ ستون) حاوی داده‌های کاربر هستند $(87 \times 9 \times 8 \times 8000 = 50.112 \text{ Mbps})$ ، که به آنها SPE (**بستهٔ کاری سنکرون** - Synchronous Payload Envelope) گفته می‌شود. اما داده‌های کاربر همیشه از سطر ۱ ستون ۴ شروع نمی‌شوند - SPE می‌تواند از هر کجای فریم شروع شود. اشاره‌گر نقطهٔ شروع SPE در اولین سطر از سرآیند خط نوشته می‌شود. اولین ستون SPE **سرآیند مسیر** (path overhead) است - که سرآیندیست برای پروتکلهای نقطه-به-نقطه.

با این تمهید (شروع SPE از هر نقطهٔ فریم SONET ، و امکان ادامهٔ آن در فریمهای بعدی) انعطاف‌پذیری سیستم بسیار بالا می‌رود (شکل ۲-۳۶ را ببینید). برای مثال، اگر پس از شروع یک فریم SONET (و ارسال مقداری اطلاعات ساختگی) داده‌های واقعی از راه برسند، می‌توان بجای انتظار تا شروع فریم بعدی، آنها را از همان نقطه وارد فریم کرد.

سلسله مراتب مالتی‌پلکس SONET را در شکل ۲-۳۷ ملاحظه می‌کنید؛ نرخها از STS-1 تا STS-192 تعریف شده‌اند. هر STS-*n* یک کاریر نوری متناظر بنام OC-*n* دارد، که بیت به بیت با آن یکسان است (باستثنای یک جابجایی کوچک که برای سنکرون شدن لازم است). نامگذاری در استاندارد SDH متفاوت است و از OC-3 شروع می‌شود، چون در سیستمهای CCITT نرخی نزدیک به 51.84 Mbps وجود ندارد. علت وجود کاریر OC-9 هم وجود یک خط اصلی پُرسرعت در ژاپن است، که چنین سرعتی دارد. از کاریرهای OC-9 و OC-18

SONET		SDH	(Mbps) سرعت داده		
الکتریکی	نوری	نوری	ناخالص	SPE	کاربر
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912

شکل ۲-۳۷. نرخهای مالتی‌پلکس SONET و SDH .

در ژاپن استفاده می‌شود. نرخ دادهٔ ناخالص شامل تمام سرآیندها می‌شود؛ در نرخ دادهٔ SPE سرآیندهای قسمت و خط حذف شده‌اند؛ و در نرخ دادهٔ کاربر تمام سرآیندها حذف، و فقط ۸۶ ستون محاسبه شده است.

اگر یک کاریر مالتی‌پلکس نشده و فقط شامل داده‌های یک منبع باشد، یک حرف *c* (بمعنای پیوسته) به انتهای نام آن اضافه می‌شود. برای مثال، OC-3 نشاندهندهٔ یک کاریر 155.52-Mbps است که از مالتی‌پلکس شدن سه خط OC-1 حاصل شده، ولی OC-3c استریمی است از یک منبع واحد با نرخ 155.52 Mbps . سه استریم OC-1در یک استریم OC-3c بصورت ستون-در-میان چیده می‌شوند: ستون ۱ از استریم ۱، سپس ستون ۱ از استریم ۲، سپس ستون ۱ از استریم ۳، بدنبال آن ستون ۲ از استریم ۱، و الی آخر - که بدین‌ترتیب فریمی مرکب از ۲۷۰ ستون و ۹ سطر بوجود می‌آید.

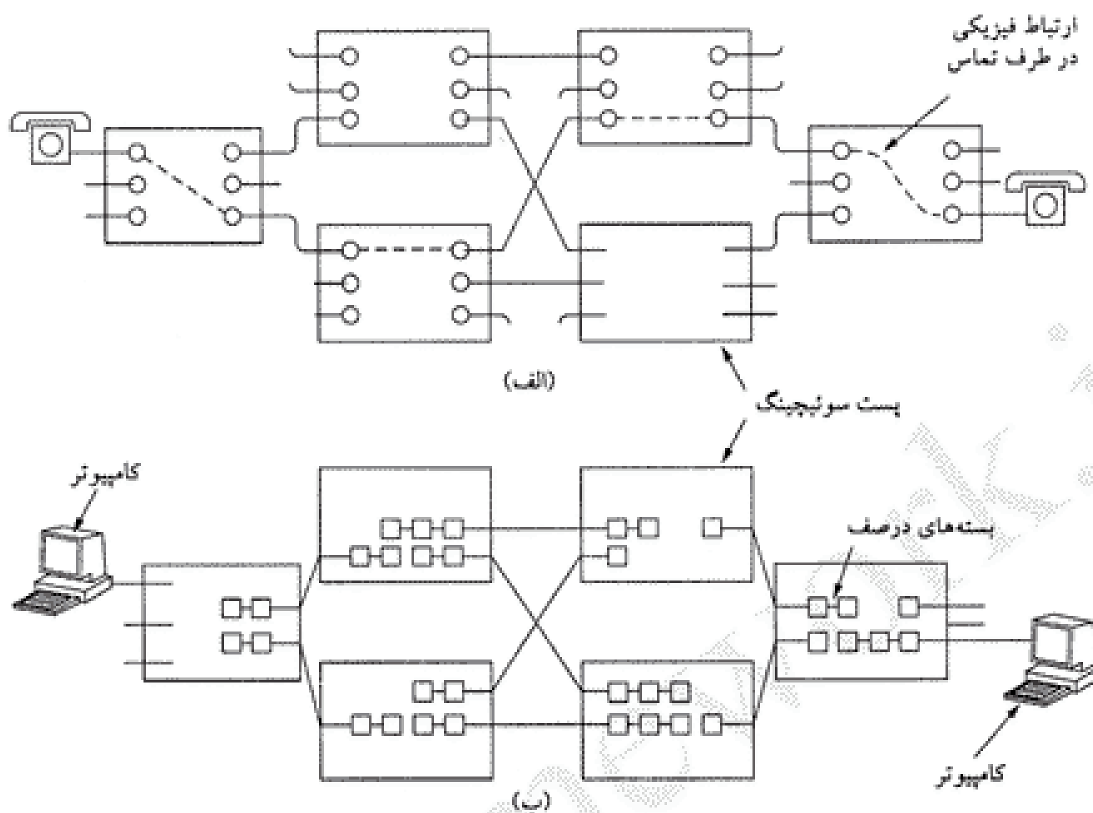
۲-۵-۵ سوئیچینگ

از دید اکثر مهندسان مخابرات، شبکهٔ تلفن دو بخش عمده دارد: خارجی (مدارهای پایانی و ترانک‌ها - تجهیزاتی که در فضای باز و بیرون مرکز تلفن نصب می‌شوند) و داخلی (سوئیچها - تجهیزاتی که در فضای بسته و داخل مرکز تلفن نصب می‌شوند). تا اینجا با بخش خارجی آشنا شدیم؛ اکنون وقت آنست که نگاهی به داخل مرکز تلفن بیندازیم.

امروزه از دو تکنیک متفاوت سوئیچینگ استفاده می‌شود: **سوئیچینگ مداری** (circuit switching) و **سوئیچینگ بسته‌ای** (packet switching) . ابتدا هر دو تکنیک را مختصراً معرفی می‌کنیم، و سپس مفصلاً به سوئیچینگ مداری (که تکنیک اصلی شبکهٔ تلفن است) می‌پردازیم. در فصلهای آینده دربارهٔ سوئیچینگ بسته‌ای بیشتر صحبت خواهیم کرد.

سوئیچینگ مداری

وقتی یک تماس تلفنی می‌گیرید، دستگاههای سوئیچینگ سیستم تلفن در صدد یافتن یک مسیر فیزیکی بین شما و تلفن مقصد برمی‌آیند. به این تکنیک که آنرا در شکل ۲-۳۸ (الف) ملاحظه می‌کنید، **سوئیچینگ مداری** گفته می‌شود. هر یک از مستطیلهایی که در این شکل می‌بینید، یک مرکز سوئیچینگ (شهری، بین‌شهری، و غیره) است. در این مثال، هر مرکز سوئیچینگ سه خط ورودی و سه خط خروجی دارد. وقتی تماس تلفنی از یکی از این مراکز سوئیچینگ می‌گذرد، یک ارتباط فیزیکی بین آن خط ورودی و یکی از خطوط خروجی برقرار می‌شود، که در این



شکل ۲-۳۸. (الف) سوئیچینگ مداری. (ب) سوئیچینگ بسته‌ای.

شکل با خط‌چین نشان داده شده است.

در روزهای اولیهٔ تلفن، این ارتباط توسط اپراتور و بکمک یک سیم فنری که پریز ورودی را به خروجی متصل می‌کرد، انجام می‌شد. اختراع دستگاه سوئیچینگ خودکار تلفن داستان جالبی دارد: این دستگاه در قرن نوزدهم در ایالت میسوری توسط فردی بنام آلمون ب. استراوگر، که شغل وی کفن و دفن بود، اختراع شد. در آن روزها وقتی کسی می‌مُرد، یکی از بازماندگان وی با اپراتور تلفن شهر تماس می‌گرفت و می‌گفت، «لطفاً مرا به یک مؤسسهٔ کفن و دفن وصل کنید.» در شهر آقای استراوگر دو مؤسسهٔ کفن و دفن وجود داشت، و از شانس بد این آقا اپراتور تلفن همسر رقیب بود. آقای استراوگر خیلی زود دریافت که اگر می‌خواهد ورشکست نشود، باید یک دستگاه سوئیچینگ خودکار تلفن اختراع کند - و این کار را کرد. همهٔ آنهایی که در سراسر دنیا با دستگاههای سوئیچینگ خودکار تلفن سروکار دارند، آنها را با نام **«دستگاه استراوگر»** می‌شناسند. (تاریخ نمی‌گوید آیا این خانم بعد از بیکار شدن توانست شغلی مانند اپراتور اطلاعات تلفن - که باید به سؤالاتی از قبیل «لطفاً شماره یک مؤسسهٔ کفن و دفن را بدهید.» پاسخ دهد - بدست آورد یا خیر؟)

البته شکل ۲-۳۸ (الف) بسیار ساده شده است، چون مسیر فیزیکی بین دو تلفن می‌تواند از لینکهای مایکروویو یا فیبر نوری (که هزاران تماس تلفنی روی آنها مالتی‌پلکس می‌شود) عبور کند. با این حال مفهوم کلی آن همچنان معتبر است: وقتی تماس تلفنی برقرار می‌شود، یک مسیر فیزیکی بین دو دستگاه تلفن بوجود می‌آید که تا آخر تماس باقی می‌ماند.

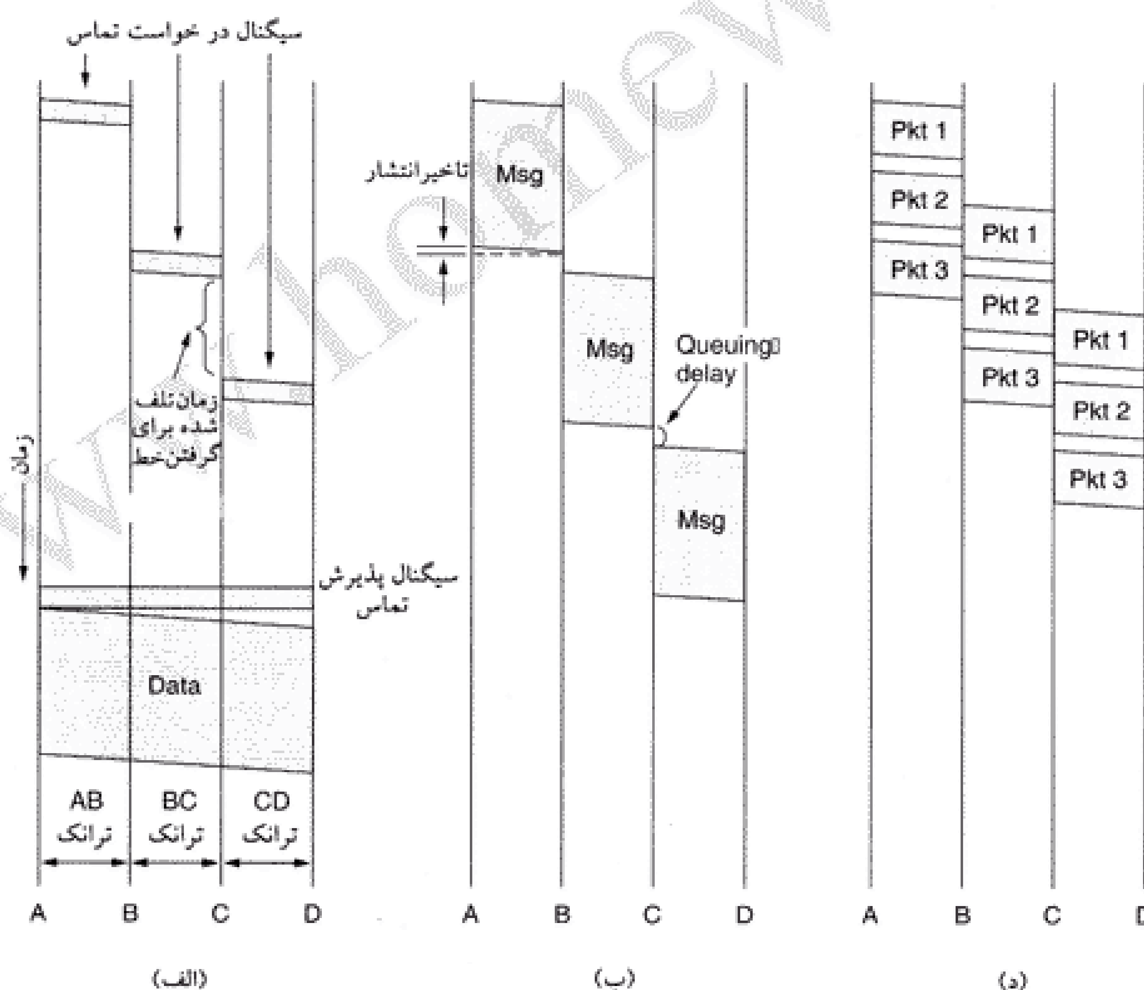
در شکل ۲-۳۸ (ب) روش جایگزین سوئیچینگ مداری، که **سوئیچینگ بسته‌ای** نام دارد، را می‌بینید. در این تکنولوژی هر بسته مستقلاً فرستاده می‌شود، بدون آنکه از قبل مسیری به آن اختصاص داده شده باشد؛ این بر عهدهٔ هر بسته است که راه خود را به سمت مقصد پیدا کند.

مهمترین ویژگی سوئیچینگ بسته‌ای اینست که قبل از ارسال هر گونه داده‌ای، بایستی یک مسیر نقطه-به-نقطه بین مبدأ و مقصد برقرار شده باشد. فاصلهٔ زمانی بین شماره‌گیری در مبدأ و زنگ خوردن تلفن مقصد براحتی می‌تواند به ۱۰ ثانیه برسد (که این زمان در تماسهای راه‌دور و بین‌المللی حتی بیشتر است). در این مدت سیستم تلفن بدنبال یافتن یک مسیر مناسب است (شکل ۲-۳۸ (الف) را ببینید). توجه داشته باشید که قبل از شروع ارسال داده، سیگنال **درخواست** (request) باید تمام مسیر را تا مقصد طی کرده، و **تصدیق** (acknowledgement) آن بازگردد. این تأخیر در بسیاری از کاربردهای کامپیوتری (مانند بررسی اعتبار مشتری در خریدهای اعتباری) قابل قبول نیست.

اما بمحض آنکه مدار برقرار شد، دیگر تأخیر چندانی بین گیرنده و فرستنده وجود ندارد (فقط زمان تأخیر انتشار امواج الکترومغناطیس، که آن هم چیزی در حدود 5 μsec/km است). دیگر اینکه (بدلیل وجود مدار اختصاصی)، در سوئیچینگ مداری پدیده‌ای بنام **ازدحام** (congestion) وجود ندارد (البته بدلیل اینکه ظرفیت خطوط و مراکز سوئیچینگ نامحدود نیست، قبل از برقراری مدار همیشه احتمال شنیدن بوق اشغال وجود دارد).

سوئیچینگ پیام

یکی دیگر از اشکال سوئیچینگ، که آنرا در شکل ۲-۳۹ (ب) ملاحظه می‌کنید، **سوئیچینگ پیام**



شکل ۲-۳۹. همزمانی رویدادها در (الف) سوئیچینگ مداری، (ب) سوئیچینگ پیام، (ج) سوئیچینگ بسته‌ای.

(message switching) است. در این نوع از سوئیچینگ نیز مسیر فیزیکی ثابتی بین فرستنده و گیرنده وجودندارد. وقتی فرستنده یک بلوک از داده‌ها را ارسال می‌کند، این داده‌ها در اولین مرکز سوئیچینگ (که همان راوتر است) **ذخیره شده** (store) ، و سپس به مرکز بعدی **هدایت** (forward) می‌شود. هر بلوک ابتدا بطور کامل دریافت شده، از نظر خطا بررسی و سپس ارسال می‌شود. همانطور که در فصل قبل هم گفتیم، به شبکه‌ای که با این روش کار می‌کند شبکهٔ **ذخیره-هدایت** (store-and-forward) گفته می‌شود.

اولین تجهیزات مخابرات الکترومغناطیسی تجهیزات سوئیچینگ پیام بودند، که برای ارسال تلگرام بکار گرفته شدند. پیام ابتدا در ادارهٔ تلگراف روی نوارهای کاغذی سوراخ می‌شد، و بعد از خوانده شدن توسط دستگاههای خاص به مرکز بعدی فرستاده می‌شد، که در آنجا بصورت یک نوار کاغذی سوراخ‌شده از دستگاه بیرون می‌آمد. اپراتور مسئول دستگاه کاغذ را پاره کرده، و در یک دستگاه **نوارخوان** (tape reader) می‌گذاشت تا بتواند پیام را بخواند (هر خط مخابراتی یک دستگاه نوارخوان داشت). این هم در واقع نوعی سوئیچینگ است، که به **سوئیچینگ نوار پاره** (torn tape switching) معروف بود. نوارهای کاغذی و سوئیچینگ پیام مدتهاست که از دور خارج شده‌اند، و ما هم بیش از این دربارهٔ آنها صحبت نخواهیم کرد.

سوئیچینگ بسته‌ای

اندازهٔ پیام در شبکهٔ سوئیچینگ پیام هیچ محدودیتی ندارد، و این به آن معناست که دستگاههای مسیریاب برای نگهداری پیامها به وسایل ذخیره‌سازی (از قبیل، دیسک) نیاز دارند. پیامد دیگر این روش آنست که یک پیام واحد می‌تواند آنقدر بزرگ باشد که برای دقایقی خط مسیریاب-مسیریاب را اشغال کند، که بدین‌ترتیب کاربرد سوئیچینگ پیام را در ارتباطات **تعاملی** (interactive) بشدت محدود می‌کند. برای حل این مشکل، **سوئیچینگ بسته‌ای** (packet switching) اختراع شد. در شبکه‌های سوئیچینگ بسته‌ای روی اندازهٔ بسته‌ها محدودیت شدیدی اعمال می‌شود، و بهمین دلیل مسیریاب‌ها نیازی به دیسک برای ذخیره کردن بسته‌ها ندارند، و می‌توانند آنها در در حافظهٔ اصلی خود ذخیره کنند. با محدود کردن اندازهٔ بسته‌ها، و اطمینان از اینکه یک کاربر نمی‌تواند خط انتقال را برای مدتی طولانی - که البته در اینجا منظور از طولانی بیش از چند میلی‌ثانیه است - به انحصار خود در آورد، شبکه‌های سوئیچینگ بسته‌ای برای کاربردهای تعاملی بسیار مناسبند. با مقایسهٔ شکلهای ۲-۳۹(ب) و (ج) یکی دیگر از مزایای روش سوئیچینگ بسته‌ای بر سوئیچینگ پیام را مشاهده می‌کنید: بستهٔ اول یک پیام چندبسته‌ای می‌تواند حتی قبل از رسیدن بستهٔ دوم به مسیریاب بعدی فرستاده شود، که این زمان تأخیر را کاهش داده و کارایی سیستم را بالا می‌برد. به دلایل فوق، شبکه‌های کامپیوتری اغلب از سوئیچینگ بسته‌ای استفاده می‌کنند؛ سوئیچینگ مداری نیز در موارد خاصی بکار برده می‌شود، ولی سوئیچینگ پیام هیچ کاربردی در شبکه‌های کامپیوتری ندارد.

سوئیچینگ مداری و سوئیچینگ بسته‌ای تفاوتهای زیادی با یکدیگر دارند. برای مثال، در یک شبکهٔ سوئیچینگ مداری قبل از ارسال اطلاعات بایستی مدار فیزیکی بین فرستنده و گیرنده برقرار شده باشد، در حالیکه در شبکه‌های سوئیچینگ بسته‌ای چنین الزامی وجود ندارد و ارسال بسته‌ها می‌تواند بلافاصله شروع شود.

پیامد لزوم برقراری مدار ثابت در سوئیچینگ مداری، اختصاص پهنای باند در تمام طول مسیر بین فرستنده و گیرنده است: تمام بسته‌ها باید از این مسیر عبور کنند. از طرف دیگر وقتی تمام بسته‌ها مجبور به عبور از یک مسیر باشند، نمی‌توانند خارج از ترتیبی که ارسال شده‌اند، به مقصد برسند. در سوئیچینگ بسته‌ای هیچ مسیر ثابتی وجود ندارد، و بسته‌ها می‌توانند از هر مسیری که (در آن لحظهٔ خاص) در شبکه موجود است عبور کنند، و حتی خارج از نظم و ترتیب اولیه به مقصد برسند.

ویژگی تحمل خطا در شبکه‌های سوئیچینگ بسته‌ای بسیار بهتر از شبکه‌های سوئیچینگ مداری است - و در واقع دلیل اختراع آن هم همین بوده است. وقتی در شبکهٔ سوئیچینگ بسته‌ای یک مسیریاب از کار می‌افتد، بسته‌ها می‌توانند از مسیرهای دیگری که وجود دارد، استفاده کرده و مسیریاب مرده را دور بزنند.

البته وجود یک پهنای باند اختصاصی در شبکه‌های سوئیچینگ مداری این مزیت را دارد که بسته‌ها بمحض رسیدن به یک مسیریاب به مسیریاب بعدی فرستاده می‌شود، و زمان تأخیر ارسال بشدت کاهش می‌یابد؛ در حالیکه در شبکه‌های سوئیچینگ بسته‌ای چنین پهنای باندی اختصاصی وجود ندارد، و بسته‌ها باید تا رسیدن نوبت ارسال در صف منتظر بمانند.

وجود مدار اختصاصی در شبکه‌های سوئیچینگ مداری بدان معناست که (بعد از برقرای مدار) دیگر حالت ازدحام - انتظار برای باز شدن راه - بروز نخواهد کرد. البته همیشه این احتمال وجود دارد که در شروع ارتباط بدلیل شلوغی شبکه، امکان اختصاص مدار وجود نداشته باشد: این نوع دیگری از ازدحام - انتظار برای تخصیص مدار - است.

پهنای باندی که به یک کاربر تخصیص داده می‌شود، در تمام مدت در اختیار وی است، حتی اگر هیچ چیز برای ارسال نداشته باشد. امکان استفاده از این مدار برای کاربران دیگر وجود ندارد. در شبکه‌های سوئیچینگ بسته‌ای اتلاف پهنای باند به شکل فوق وجود ندارد، و بهمین دلیل کارایی کلی آن بسیار بهتر است. درک این تفاوت بین سوئیچینگ بسته‌ای و سوئیچینگ مداری بسیار مهم است: تفاوت تضمین سرویس به قیمت اتلاف منابع، با استفادهٔ بهینه از منابع به قیمت عدم تضمین سرویس.

سوئیچینگ بسته‌ای از تکنیک ذخیره-هدایت استفاده می‌کند. در این روش هر بسته قبل از ارسال به مسیریاب بعدی باید بطور کامل دریافت و در حافظهٔ مسیریاب ذخیره شود. این روش تأخیر نسبتاً قابل ملاحظه‌ای ایجاد می‌کند؛ در حالیکه در سوئیچینگ مداری، بیت‌ها بطور پیوسته روی مدار منقل می‌شوند و چنین تأخیری وجود ندارد.

تفاوت دیگر اینست که سوئیچینگ مداری بطور کامل شفاف است: فرستنده و گیرنده می‌توانند از هر نرخ بیت، فرمت، و یا روش فریم‌بندی که می‌خواهند استفاده کنند؛ کاریر در این مورد هیچ چیز نمی‌داند، و به آن اهمیتی هم نمی‌دهد. اما در سوئیچینگ بسته‌ای این کاریر است که پارامترهای اصلی را تعیین می‌کند. تفاوت این دو تقریباً مانند جاده و راه‌آهن است: در جاده این مسافر است که سرعت، اندازه و نوع وسیلهٔ نقلیه را انتخاب می‌کند، در حالیکه در راه‌آهن انتخاب این پارامترها بر عهدهٔ شرکت مسافربری (کاریر) است. همین شفافیت است که به سیستم تلفن اجازه می‌دهد انواع اطلاعات (صوت، فکس و داده) را منتقل کند.

آخرین تفاوت سوئیچینگ مداری و سوئیچینگ بسته‌ای روش محاسبهٔ هزینه است. در سوئیچینگ مداری (بدلایل تاریخی) هزینه بر اساس مسافت و مدت محاسبه می‌شود. در تلفنهای همراه، مسافت (البته باستثنای مکالمات بین‌المللی) نقشی ندارد، و مدت مکالمه نیز نقش ناچیزی دارد (برای مثال، تفاوت هزینهٔ مکالمه در روز، شب یا ایام تعطیل). در سوئیچینگ بسته‌ای، اساساً چیزی بنام مدت مکالمه وجود ندارد، و فقط گاهی حجم ترافیک نقشی‌در هزینه‌ها بازی می‌کند. در مصارف خانگی، معمولاً هزینه‌ها بصورت ماهیانهٔ ثابت اخذ می‌شود، چون این روش برای ISP ها ساده‌تر است و کاربران نیز راحتتر با آن کنار می‌آیند، ولی کاریرهای اصلی شبکه هزینه‌ها را بر اساس حجم ترافیک دریافت می‌کنند. تفاوتهایی راکه در این قسمت برشمردیم، بصورت خلاصه در شکل ۲-۴۰ ملاحظه می‌کنید.

سوئیچینگ مداری و بسته آنقدر مهم هستند که بزودی دوباره به این مبحث برگشته، و تکنولوژیهای مختلف آنها را به تفصیل بررسی خواهیم کرد.

سوئیچینگ بسته‌ای	سوئیچینگ مداری	آیتم
لازم ندارد	لازم دارد	برقراری تماس
خیر	بلی	مسیر فیزیکی اختصاصی
خیر	بلی	تمام بسته ها از یک مسیر عبور می کنند
خیر	بلی	بسته ها به ترتیب دریافت می‌شوند
خیر	بلی	خرابی سوئیچ مرگ‌آور است
متغیر	ثابت	پهنای باند موجود
در هربسته	در لحظه شروع	زمان ازدحام احتمالی
خیر	بلی	پهنای باند تلف شده
بلی	خیر	ذخیره - هدایت
خیر	بلی	شفافیت
به ازای هر بسته	در دقیقه	هزینه

شکل ۲-۴۰. مقایسهٔ شبکه‌های سوئیچینگ مداری و سوئیچینگ بسته‌ای.

۲ـ۶ شبکهٔ تلفن همراه

سیستم تلفن معمولی (حتی اگر سرعت آن به دهها گیگابایت برسد) دستهٔ خاصی از کاربران را هرگز راضی نخواهد کرد: آنهایی که در یک جا بند نمی‌شوند. امروزه مردم می‌خواهند از هر جایی که هستند (داخل هواپیما، سوار بر اتومبیل، کنار دریا، نوک قلهٔ کوهها و یا از اعماق جنگل) تلفن بزنند - و حتماً تا چند سال دیگر انتظار دارند از این نقاط ایمیل خود را نیز چک کنند، و یا در وب گشت بزنند. پیامد این گرایش رشد چشمگیر تلفنهای غیرثابت در سالهای اخیر است. در این قسمت قصد داریم مبحث تلفن همراه را بررسی کنیم.

تلفنهای غیرثابت به دو دستهٔ بزرگ تقسیم می‌شوند: **گوشی بیسیم** (cordless phone) ، و **تلفن همراه** (mobile phone) - که گاهی به آن **تلفن سلولی** (cell phone) نیز گفته می‌شود. گوشی بیسیم عبارتست از یک دستگاه مرکزی و یک گوشی متحرک، که برای مصارف خانگی (مسافتهای کوتاه) طراحی شده است. این وسیله هرگز کاربردی در شبکه نداشته، و ما هم بیش از این به آن نخواهیم پرداخت. تلفن همراه، که امروزه کاربرد گسترده‌ای در ارتباطات صدا و داده دارد، موضوع اصلی بحث ماست.

تکامل تلفنهای همراه سه نسل را با تکنولوژیهای متفاوت پشت سر گذاشته است:

۱. صدای آنالوگ
۲. صدای دیجیتال
۳. صدای دیجیتال و داده (اینترنت، ایمیل، و غیره)

با اینکه بیشتر تکنولوژی این سیستمها مورد علاقهٔ ماست، اما جالبست بدانید که سیاست و تصمیمات اقتصادی چه تأثیری بر تکامل این سیستمها دارد. اولین سیستم تلفن همراه در ایالات متحدهٔ آمریکا و توسط AT&T اختراع شد، که FCC هم بلافاصله آنرا در تمام کشور اجباری کرد. در نتیجه، تمام ایالات متحده صاحب یک سیستم واحد تلفن همراه (آنالوگ) شد، و تلفنی که در کالیفرنیا خریده شده بود، در نیویورک هم کار می‌کرد. اما در اروپا اوضاع کاملاً عکس این بود، و هر کشوری سیستم خاص خود را طرحی کرد، که نتیجهٔ آن یک هرج و مرج کامل بود.

اروپا از اشتباه خود درس گرفت و وقتی نوبت تلفن همراه دیجیتال رسید، تمام شرکتهای مخابرات دولتی دور

هم جمع شده و بر سر یک استاندارد واحد (GSM) به توفق رسیدند، که در نتیجه تلفنهای موبایل اروپایی در تمام نقاط این قاره کار می‌کند. در همان زمان ایالات متحده به این نتیجه رسیده بود که دولت نباید در موضوع استاندارد دخالت کند، و در نتیجه سرنوشت تلفن همراه دیجیتال به بازار سپرده شد. این تصمیم باعث شد تا انواع مختلفی از تلفن همراه دیجیتال تولید و وارد بازار شود - ایالات متحده اکنون دو سیستم بزرگ تلفن همراه دیجیتال (بعلاوهٔ یک سیستم کوچکتر) دارد، که هیچکدام با دیگری سازگار نیست.

اروپا فقط در یک دورهٔ زمانی کوتاه در زمینهٔ تعداد کاربران تلفن همراه از آمریکا عقب بود، ولی اکنون از آن بسیار پیش افتاده است، که یکی از دلایل آن سیستم یکپارچهٔ تلفن همراه در اروپاست، اما دلایل دیگری هم برای آن وجود دارد. تفاوت دیگر سیستم تلفن همراه آمریکا و اروپا (که باعث سرافکندگی آمریکایی‌هاست) شماره‌های آنهاست. در ایالات متحده شماره‌های تلفن همراه با شماره تلفنهای معمولی (ثابت) مخلوط است. هیچ راهی برای تماس‌گیرنده وجود ندارد تا تشخیص دهد شماره‌ای که دارد می‌گیرد (مثلاً، 5678-234 (212))، یک شمارهٔ معمولی (کم‌هزینه) است یا یک شمارهٔ تلفن همراه (با هزینهٔ زیاد). برای عصبی‌تر کردن مردم، شرکتهای تلفن قرار گذاشته‌اند تا هزینهٔ تماسها را به پای صاحب تلفن همراه بنویسند. بهمین دلیل اغلب مردم در خرید تلفن همراه تردید دارند، و از صورتحسابهای نجومی که ممکنست (در نتیجهٔ تماس دیگران) برای آنها بیاید، می‌ترسند. در اروپا، شماره‌های تلفن همراه کُد ناحیهٔ مشخصی دارند (مانند تلفنهای 800 یا 900)، و بسادگی از تلفنهای معمولی قابل تشخیص هستند. هزینهٔ تماس هم مثل تلفنهای معمولی به پای تماس‌گیرنده نوشته می‌شود (البته باستثنای تماسهای بین‌المللی، که هزینهٔ تماس بین دو طرف تقسیم می‌شود).

دلیل دیگری که باعث پذیرش گستردهٔ تلفن همراه در اروپا شده، ابداع **تلفنهای همراه از پیش پرداخت شده** (pre-paid) است، که تا ۷۵٪ تلفنهای این قاره را در برخی نقاط شامل می‌شود. این تلفنها را می‌توان در هر فروشگاهی (بدون هیچگونه تشریفات خاص) خرید - فقط پولش را بده و استفاده کن. این تلفنها معمولاً با اعتبار ۲۰ یا ۵۰ یورو (واحد پول اروپا) عرضه می‌شوند، و بعد از صفر شدن اعتبار می‌توان آنها را (با استفاده از یک PIN code سرّی) دوباره شارژ کرد. امروزه هر نوجوان (و حتی بچه‌ای) در اروپا یک چنین تلفن همراهی دارد، و والدین وی می‌توانند بسادگی محلی او را پیدا کنند (بدون اینکه از صورتحسابهای نجومی آن بترسند). این تلفنها (البته اگر از آنها برای تماس گرفتن استفاده نشود) ماهها دوام می‌آورند، چون نه هزینهٔ ماهیانه دارند نه برای تلفنهایی که به آنها می‌شود، اعتباری کسر می‌شود.

۲-۶-۱ تلفن‌های همراه نسل اول: صدای آنالوگ

صحبت از سیاست و روشهای بازاریابی دیگر بس است؛ اجازه دهید به کار اصلی خود یعنی تکنولوژی بپردازیم. حتی از اوایل قرن بیستم نیز تلفنهای متحرک رادیو-بیسیم در زمینه‌های نظامی و مسافرتهای دریایی کارایی خود را به اثبات رسانده بودند. در سال ۱۹۴۶، اولین تلفنهای مخصوص اتومبیل در سنت‌لوئیس راه‌اندازی شدند. در این سیستم آنتنهای (فرستنده-گیرنده) بزرگی روی ساختمانهای بلند نصب شده بود، و ارسال و دریافت روی یک کانال واحد صورت می‌گرفت. صحبت کردن و شنیدن در آن واحد امکان نداشت: برای صحبت کردن باید یک دکمه را فشار می‌دادید، و با رها کردن آن دیگر نمی‌توانستید حرف بزنید. تا اوایل دههٔ ۱۹۵۰، این سیستم، که به **فشار بده-حرف بزن** (push-to-talk) معروف بود، در بسیاری از شهرهای بزرگ آمریکا نصب شد. این سیستم امروزه هم در اتومبیلهای پلیس و تاکسی تلفنی مورد استفاده قرار می‌گیرد.

در دههٔ ۱۹۶۰، IMTS (**سیستم تلفن همراه بهبودیافته** - Improved Mobile Telephone System) نصب و راه‌اندازی شد. این سیستم نیز از فرستنده-گیرنده‌های قوی (۲۰۰ واتی) در نقاط مرتفع استفاده می‌کرد، ولی برای ارسال و دریافت دو فرکانس متفاوت داشت، که بدین ترتیب نیازی به دکمهٔ «فشار بده-حرف بزن» نبود. علاوه بر

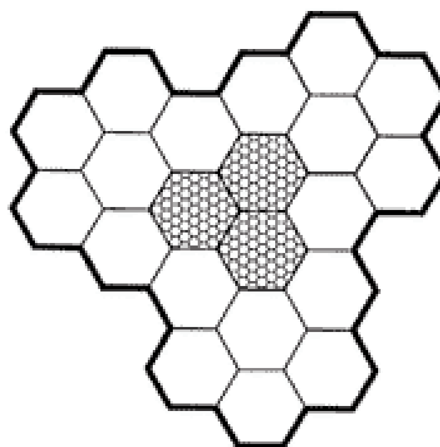
آن، برای ارسال نیز از فرکانسهای متعددی استفاده می‌شد، و خطر شنیده شدن صحبت دیگران (بر خلاف تلفنهای تاکسی تلفنی) نیز وجود نداشت.

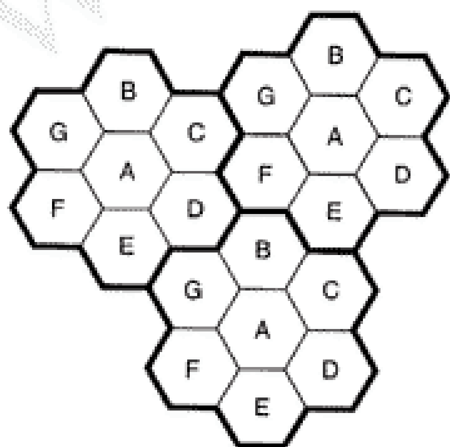
سیستم IMTS از ۲۳ کانال (در طیف 150-450 Mhz) پشتیبانی می‌کرد. بدلیل کم بودن تعداد این کانالها، کاربران مجبور بودند برای شنیدن بوق آزاد مدت زیادی صبر کنند. همچنین بدلیل قدرت زیاد فرستنده-گیرنده‌های این سیستم، تا شعاع چند صد کیلومتری در سیستمهای رادیویی اختلال ایجاد می‌کردند. همهٔ این دلایل IMTS را به سیستمی غیرعملی تبدیل کرده بود.

سیستم تلفن همراه پیشرفته

در سال ۱۹۸۲، با اختراع AMPS (**سیستم تلفن همراه پیشرفته** – Advanced Mobile Phone System) توسط شرکت Bell Labs اوضاع تغییر کرد. این سیستم در انگلستان (با نام TACS) و ژاپن (با نام MCS-L1) نیز نصب و بکار گرفته شد. با آنکه این سیستم دیگر تکنولوژی برتر محسوب نمی‌شود، اما از آنجائیکه بمنظور سازگاری با گذشته بسیاری از ویژگیهای بنیادی آن در تلفن همراه دیجیتال (DAMPS) لحاظ شده است، قدری دربارهٔ آن صحبت خواهیم کرد.

در تمام سیستمهای تلفن همراه، یک منطقهٔ جغرافیایی به تعدادی **سلول** (cell) تقسیم می‌شود – که بهمین دلیل به آن **تلفن سلولی** (cell phone) نیز می‌گویند. در AMPS هر سلول ۱۰ تا ۲۰ کیلومتر قطر دارد؛ در سیستمهای دیجیتالی این سلولها کوچکترند. هر سلول دارای تعدادی فرکانس است، که سلولهای همسایه از آنها استفاده نمی‌کنند. ایدهٔ کلیدی در سیستمهای سلولی که باعث شده تا ظرفیت آنها بسیار بیشتر از سیستمهای قبلی باشد، استفاده از سلولهای نسبتاً کوچک و بکارگیری فرکانسها در سلولهای نزدیک هم (البته، نه سلولهای مجاور) است. در حالیکه یک سیستم IMTS با قطر ۱۰۰ کیلومتر می‌تواند روی هر فرکانس یک تماس داشته باشد، یک سیستم AMPS قادر است (با جدا کردن فرکانس سلولهای مجاور) روی هر فرکانس ۱۰ تا ۱۵ تماس برقرار کند. هر چه اندازهٔ سلولها کوچکتر (و تعداد آنها بیشتر) باشد، ظرفیت سیستم بالاتر خواهد رفت. با کوچکتر شدن سلولها می‌توان قدرت فرستنده-گیرنده‌ها را نیز کمتر کرد، و از تجهیزات ساده‌تر و ارزانتری استفاده کرد. طبق مقررات FCC ، حداکثر قدرت تشعشعی دستگاه‌های تلفن همراه 0.6 وات، و فرستنده-گیرنده‌های مخصوص اتومبیل 3 وات تعیین شده است.


(ب)



(الف)

شکل ۲-۴۱. (الف) فرکانسهای سلولهای مجاور یکسان نیستند. (ب) برای افزایش تعداد کاربران، می‌توان از سلولهای کوچکتر استفاده کرد.

ایدهٔ تکرار فرکانسها را در شکل ۲-۴۱ (الف) ملاحظه می‌کنید. سلولها تقریباً به شکل دایره هستند، ولی برای سادگی کار بهتر است آنها را شش‌ضلعی فرض کنیم. در این شکل تمام سلولها هم‌اندازه‌اند، و در گروههای هفت‌تایی دسته‌بندی شده‌اند. هر حرف نشاندهندهٔ مجموعه‌ای از فرکانسها است. توجه کنید که هر مجموعهٔ فرکانسی با نزدیکترین سلول مشابه حداقل و سلول فاصله دارد، که این باعث به حداقل رسیدن تداخل فرکانسها خواهد شد.

یکی از بزرگترین معضلات سیستمهای تلفن همراه، یافتن نقاط مناسب برای نصب آنتنهای مرکزی سلولهاست. این مشکل باعث شده تا شرکتهای تلفن برای استفاده از مناره‌های رفیع کلیساها دست بدامن آنها شوند.

وقتی در یک منطقه تعداد کاربران افزایش می‌یابد و سیستم دیگر جوابگوی بار مکالمات نیست، (با کم کردن قدرت آنتنها) هر سلول به سلولهای کوچکتر (microcell) تقسیم می‌شود، تا بتوان از فرکانسها بدفعات بیشتر استفاده کرد - شکل ۲-۴۱ (ب) را ببینید. در مواقعی که تعداد زیادی تلفن همراه برای مدتی کوتاه در یک نقطه گرد می‌آیند (مانند مسابقات ورزشی و کنسرتهای موسیقی)، شرکتهای تلفن با استقرار آنتنهای متحرک (که ارتباط ماهواره‌ای دارند) موقتاً سلولهای کوچکتری ایجاد می‌کنند. اینکه اندازهٔ یک سلول چقدر باید باشد، مسئلهٔ پیچیده‌ایست که در (Hac, 1995) دربارهٔ آن بحث شده است.

در مرکز هر سلول یک ایستگاه قرار دارد که تمام تلفنهای داخل سلول امواج خود را به آن می‌فرستند. در سیستمهای کوچک، تمام این ایستگاهها به یک دستگاه مرکزی بنام MTSO (**مرکز سوئیچینگ تلفن همراه** - Mobile Telephone Switching Office) یا MSC (**مرکز سوئیچینگ همراه** - Mobile Switching Center) متصل می‌شوند. در سیستمهای بزرگتر چندین MTSO وجود دارند، که بنوبهٔ خود به یک MTSO بالاتر متصلند. MTSO ها، در واقع، همان ایستگاههای پایانی سیستم تلفن همراه هستند، که به حداقل یک ایستگاه پایانی سیستم تلفن ثابت نیز ارتباط دارند. ارتباط MTSO ها با تلفنهای همراه، با یکدیگر و با مراکز PSTN از طریق یک شبکهٔ سوئیچینگ بسته‌ای صورت می‌گیرد.

هر تلفن همراه در هر لحظه در یک سلول (و تحت کنترل ایستگاه مرکزی آن) قرار دارد. وقتی این تلفن سلول را ترک می‌کند، ایستگاه مرکزی متوجه ضعیف شدن سیگنال آن شده و از تمام ایستگاههای مجاور میزان قدرت دریافتی از این تلفن را می‌پرسد. سپس، این ایستگاه کنترل تلفن مزبور را به ایستگاهی که بیشترین قدرت را گزارش کرده (و در واقع، تلفن وارد آن شده)، تحویل می‌دهد. رئیس جدید نیز به تلفن همراه اطلاع می‌دهد که (اگر مایل به ادامهٔ مکالمه است) کانال خود را عوض کند (چون فرکانس قبلی آن در هیچیک از سلولهای همسایه وجود ندارد). این فرآیند (که به آن **پاس‌کاری** - handoff - گفته می‌شود) تقریباً 300 msec طول می‌کشد. تخصیص کانال توسط MTSO صورت می‌گیرد (چون ایستگاههای مرکزی چیزی جز رله‌های رادیویی ساده نیستند).

پاس‌کاری می‌تواند به دو طریق صورت گیرد. در **پاس‌کاری نرم** (soft handoff) تلفن همراه قبل از آن که ایستگاه قدیم آنرا قطع کند، به ایستگاه جدید متصل می‌شود. در این روش کاربر هیچ نوع قطعی احساس نمی‌کند. اما مشکل این روش آنست که دستگاه تلفن همراه باید بتواند در آن واحد خود را روی دو فرکانس (ایستگاه قبلی و ایستگاه جدید) تنظیم کند. تلفنهای همراه نسل اول و دوم هیچکدام قادر به انجام چنین کاری نیستند.

در **پاس‌کاری سخت** (hard handoff) ایستگاه قدیمی قبل از اتصال تلفن همراه به ایستگاه جدید، آنرا قطع می‌کند. اگر ایستگاه جدید به هر دلیلی (مثلاً، نداشتن باند خالی) نتواند تلفن را تحویل بگیرد، ارتباط کاربر یکباره قطع خواهد شد. این یکی از مشکلات اجتناب‌ناپذیر تلفنهای همراه نسل اول و دوم است.

کانال‌ها

سیستم AMPS دارای ۸۳۲ کانال دو-طرفهٔ همزمان است، که هر کانال خود از دو کانال یکطرفهٔ ساده تشکیل می‌شود (۸۳۲ کانال دریافت و ۸۳۲ کانال ارسال). کانالهای یکطرفهٔ دریافت روی فرکانسهای 824-849 MHz و کانالهای یکطرفهٔ ارسال روی فرکانسهای 869-894 MHz کار می‌کنند (پهنای باند هر کانال 30 kHz است). همانطور که می‌بینید، AMPS از FDM برای تقسیم پهنای باند استفاده می‌کند.

امواج در فرکانس 800 MHz طول موجی در حدود 40 cm دارند، و به خط مستقیم حرکت می‌کنند. درختان و گیاهان این امواج را جذب می‌کنند، و در برخورد با زمین نیز منعکس می‌شوند. موجی که به ایستگاه مرکزی سلول می‌رسد، می‌تواند مستقیماً از گوشی تلفن همراه آمده باشد، و یا انعکاس آن از سطح زمین یا ساختمانها باشد، که این می‌تواند باعث **پژواک** (echo) یا **محوشدگی چندمسیره** (multipath fading) شود. گاهی حتی امکان شنیدن مکالماتی که در مسافتی طولانی چندین بار به سطح زمین خورده و منعکس شده، نیز وجود دارد.

۸۳۲ کانال AMPS به چهار دسته تقسیم می‌شوند:

۱. کانالهای کنترل (ایستگاه به تلفن همراه) برای مدیریت سیستم
۲. کانالهای فراخوانی (ایستگاه به تلفن همراه) برای اعلام اینکه مکالمه‌ای در انتظار کاربر است
۳. کانالهای دسترسی (دو طرفه) برای برقراری مکالمات و تخصیص کانال
۴. کانالهای داده (دو طرفه) برای صدا، فکس، یا داده

تعداد کانالهای کنترل ۲۱ عدد است، که بطور ثابت در PROM تلفنهای همراه نوشته شده است. از آنجائیکه استفاده از فرکانسهای مشابه در سلولهای مجاور مجاز نیست، تعداد کانالهای قابل استفاده در هر سلول بسیار کمتر از ۸۳۲ (و در واقع، چیزی نزدیک به ۴۵ کانال) است.

مدیریت مکالمه

هر تلفن همراه در سیستم AMPS دارای یک شماره سریال ۳۲ بیتی و یک شماره تلفن ۱۰ رقمی است، که در PROM آن نوشته شده است. شمارهٔ تلفن همراه دارای یک کُد ناحیهٔ ۳ رقمی (که بصورت ۱۰ بیتی کُد شده) و یک شمارهٔ مشترک ۷ رقمی (که بصورت ۲۴ بیتی کُد شده) - مجموعاً ۳۴ بیت - است.

وقتی تلفن روشن می‌شود، ۲۱ کانال کنترل از پیش برنامه‌ریزی شده را اسکن می‌کند، تا قویترین سیگنال را پیدا کند. سپس، شماره سریال ۳۲ بیتی و شماره تلفن ۳۴ بیتی خود را منتشر می‌کند. تمام اطلاعات کنترلی در AMPS دیجیتال هستند (برخلاف کانالهای صدا، که آنالوگ می‌باشند)، و این اطلاعات به دفعات و همراه با کُدهای تصحیح خطا منتشر می‌شوند.

وقتی ایستگاه مرکزی سلول این اعلام را دریافت کرد، آنرا به MTSO می‌فرستد، که آن هم (پس از ثبت کاربر جدید) محل وی را به نزدیکترین MTSO اعلام می‌کند. در حالت عادی، یک تلفن همراه هر ۱۵ دقیقه خود را به ایستگاه مرکزی معرفی می‌کند.

برای برقراری یک تماس، کاربر (بعد از روشن کردن تلفن) شمارهٔ مورد نظر را وارد کرده، و دکمهٔ CALL را فشار می‌دهد. تلفن این شماره را به همراه کُد شناسایی خود روی یکی از کانالهای دسترسی (access channel) به ایستگاه مرکزی می‌فرستد. (اگر تداخلی پیش آید، تلفن این عملیات را بعداً تکرار می‌کند.) وقتی ایستگاه مرکزی سلول این درخواست را دریافت کرد، به MTSO اطلاع می‌دهد. اگر تماس‌گیرنده یکی از مشترکین آن MTSO باشد، MTSO بدنبال یک کانال خالی می‌گردد تا به وی تخصیص دهد. اگر کانال خالی موجود بود، شمارهٔ آن روی کانال کنترل به تلفن برگشت داده می‌شود. با گرفتن کانال دسترسی از MTSO ، تلفن همراه بطور خودکار به آن

کانال سوئیچ کرده و منتظر می‌ماند تا طرف مقابل گوشی را بردارد.

تماسهای ورودی به طریق دیگری عمل می‌کنند. تلفنهایی که بیکار هستند، دائماً به **کانال فراخوانی** (paging channel) گوش می‌کنند، تا پیامهایی را که برای آنان می‌رسد دریافت کنند. وقتی یک شمارهٔ تلفن همراه گرفته می‌شود (خواه از یک تلفن ثابت یا یک تلفن همراه دیگر)، یک بسته به MTSO ی آن تلفن ارسال می‌شود تا محل وی را پیدا کند. MTSO نیز که محل تمام مشترکین فعال خود را می‌داند، یک بسته به ایستگاه مرکزی آن تلفن می‌فرستد، که آن هم بنوبهٔ خود یک پیام (با مضمون: «تلفن 14 ، تو آنجایی؟») روی کانال فراخوانی منتشر می‌کند. تلفن مقصد هم با ارسال پاسخ "Yes" روی کانال دسترسی جواب می دهد. وقتی ایستگاه مرکزی پاسخ "Yes" را دریافت کرد، با پیامی مانند این عکس‌العمل نشان می‌دهد: «تلفن 14 ، روی کانال 3 به تماس جواب بده.» در اینجا، تلفن همراه به کانال 3 سوئیچ کرده، و شروع به زنگ زدن (یا نواختن یکی از ملودیهای عجیب و غریبی که این روزها همه جا شنیده می‌شود) می‌کند.

۲-۶-۲ تلفن‌های همراه نسل دوم: صدای دیجیتال

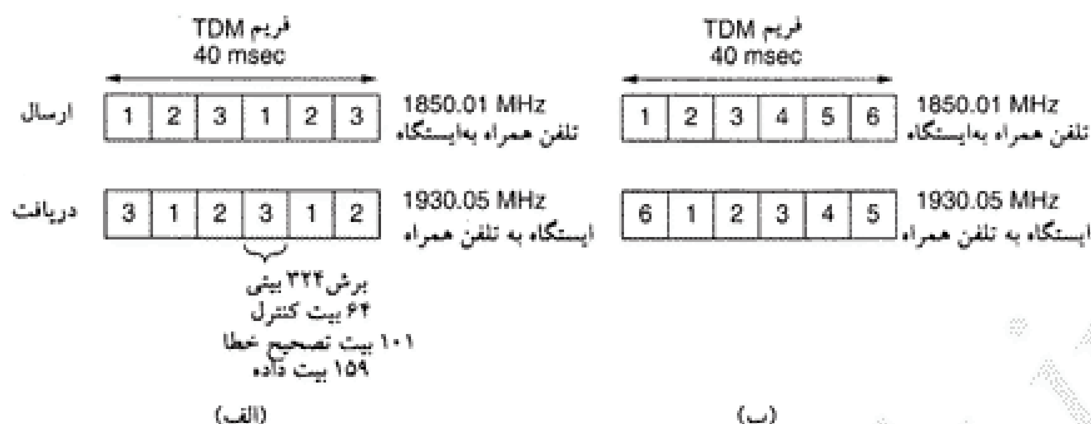
تلفن همراه نسل اول آنالوگ بود، و تلفن همراه نسل دوم دیجیتال. همانطور که نسل اول استانداردی جهانی نداشت، نسل دوم هم چنین استاندارد جهانشمولی ندارد. امروزه چهار سیستم تلفن همراه نسل دوم مشغول بکار هستند: D-AMPS ، GSM ، CDMA و PDC . در این قسمت سه سیستم اول را بررسی خواهیم کرد؛ PDC فقط در ژاپن راه‌اندازی شده، و در واقع همان D-AMPS است که برای سازگاری با سیستمهای نسل اول آنالوگ ژاپن تغییراتی در آن صورت گرفته است. اصطلاح PCS (**سرویسهای مخابرات شخصی** - Personal Communications Services) از لحاظ فنی به معنای تلفن همراه دیجیتال با فرکانس 1900 MHz است، اما امروزه این کلمه عمدتاً به سیستمهای نسل دوم (یعنی، دیجیتال) تعبیر می‌شود.

D-AMPS : سیستم تلفن همراه پیشرفتهٔ دیجیتال

نسل دوم سیستمهای AMPS (که کاملاً دیجیتال است) D-AMPS (**سیستم تلفن همراه پیشرفتهٔ دیجیتال** - Digital Advanced Mobile Phone System) نام دارد. این سیستم تحت استاندارد بین‌المللی IS-54 (و خلف آن تحت استاندارد IS-136) تعریف شده است. سیستم D-AMPS بگونه‌ای طراحی شده که با AMPS سازگاری کامل داشته باشد، بهمین دلیل تلفنهای نسل اول و دوم می‌توانند همزمان در یک سلول کار کنند. D-AMPS از همان کانالهای 30 kHz نسل اول (و دقیقاً با همان فرکانسها) استفاده می‌کند، بنابراین در یک سلول یک کانال می‌تواند آنالوگ کار کند، در حالیکه کانال مجاور در حال کار بصورت دیجیتال است. این MTSO ی مسئول سلول است که (بر اساس نسبت تلفنهای نسل اول و دوم در هر سلول) تعیین می‌کند کدام کانالها آنالوگ و کدام کانالها دیجیتال کار کنند - و (با تغییر این نسبت) می‌تواند ترکیب کانالها را بصورت دینامیک تغییر دهد.

بعد از عملیاتی شدن سرویسهای D-AMPS و بمنظور پاسخگویی به تقاضای رو به فزونی بازار، باند فرکانسی جدیدی به آن تخصیص داده شد. کانالهای **دریافت از کاربر** (upstream) در محدودهٔ 1850-1910 MHz ، و کانالهای **ارسال به کاربر** (downstrean) در محدودهٔ 1930-1990 MHz هستند. طول موج در این باند فقط 16 cm است، بنابراین آنتن استاندارد (**آنتن یکچهارم طول موج**) تلفنهایی که در این باند کار می‌کنند، فقط 4 cm خواهد بود، و بهمین دلیل این تلفنها می‌توانند بسیار کوچکتر ساخته شوند. با این حال، اغلب تلفنهای D-AMPS هر دو باند فرکانسی (850 MHz و 1900 MHz) را پشتیبانی می‌کنند.

در یک تلفن همراه D-AMPS ، سیگنال خروجی میکروفون بعد از دیجیتایز شدن با استفاده از مدلهایی که بسیار بهینه‌تر از **مدولاسیون دلتا** (delta modulation) **و کُد کردن پیشگویانه** (predictive encoding) هستند،



شکل ۲-۴۲. (الف) یک کانال D-AMPS با سه کاربر. (ب) یک کانال D-AMPS با شش کاربر.

فشرده می‌شود. این مدلهای فشرده‌سازی با استفاده از خصوصیات دستگاه صوتی انسان، پهنای باند لازم را از 56-kbps به 8 kbps یا کمتر تقلیل می‌دهند. این کار که توسط دستگاهی بنام vocoder انجام می‌شود (Bellamy, 2000 را ببینید)، در دستگاه تلفن صورت می‌گیرد نه در ایستگاه مرکزی، که بدین ترتیب تعداد بیت‌های منتقل شده روی لینک هوایی کاهش خواهد یافت. در تلفنهای ثابت این تکنیک هیچ مزیتی بدنبال ندارد، چون کاستن از ترافیک مدار پایانی هیچ تأثیری روی ظرفیت کل سیستم نخواهد داشت.

فشرده کردن صدای دیجیتال تلفنهای همراه (همان کاری که در D-AMPS انجام می‌شود) مزیت فوق‌العاده‌ای بهمراه دارد، چون می‌توان با استفاده از **مالتی‌پلکس تقسیم زمانی** (TDM) یک زوج فرکانسی (ارسال و دریافت) را بین سه کاربر به اشتراک گذاشت. هر زوج فرکانسی از 25 frame/sec (هر فریم معادل 40 msec) پشتیبانی می‌کند. هر فریم نیز بنوبهٔ خود به شش بُرش زمانی 6.67 msec تقسیم می‌شود (شکل ۲-۴۲ (الف) را ببینید).

هر فریم به سه کاربر سرویس (ارسال و دریافت) می‌دهد، که به نوبت از آن استفاده می‌کنند. برای مثال در بُرش زمانی 1 در شکل ۲-۴۲ (الف)، کاربر ۱ می‌تواند در حال ارسال به ایستگاه مرکزی باشد، در حالیکه در همان زمان کاربر ۳ در حال دریافت است. هر بُرش زمانی طولی معادل 324 بیت دارد، که 64 بیت آن برای مصارف کنترلی اختصاص یافته، و بقیهٔ 260 بیت در اختیار کاربر است. از این 260 بیت، 101 بیت برای کُدهای تصحیح خطا (که در لینکهای هوایی پُرنویز بسیار هم لازمند) بکار می‌رود، که بدین ترتیب فقط 159 بیت برای انتقال صدا باقی می‌ماند. با احتساب ۵۰ بُرش زمانی در هر ثانیه، پهنای باند موجود برای صدای فشرده فقط 8 kbps ($\frac{۱}{۷}$ پهنای باند PCM) است.

با استفاده از الگوریتمهای فشرده‌سازی بهتر، می‌توان پهنای باند لازم را حتی به 4 kbps کاهش داد، تا شش کاربر بتوانند در آن واحد از یک فریم استفاده کنند (شکل ۲-۴۲ ب). از دید شرکتهای تلفن، توانایی فشرده کردن سه یا شش کاربر D-AMPS در کانالی که فقط یک کاربر AMPS می‌تواند از آن استفاده کند، یک بُرد واقعی است و دلیل محبوبیت PCM نزد شرکتهای تلفن همراه نیز همین است. البته کیفیت صدای 4 kbps هرگز به پای صدای 56 kbps نمی‌رسد، اما شرکتهای تلفن همراه هم کمتر روی کیفیت صدای سرویسهای خود تبلیغ می‌کنند. در مورد سرویسهای داده، کیفیت یک خط 8 kbps حتی با مودمهای 9600-bps نیز قابل مقایسه نیست.

ساختار کنترلی D-AMPS نسبتاً پیچیده است: هر ۱۶ فریم تشکیل یک **ابَرفریم** (superframe) می‌دهند، که اطلاعات کنترلی بدفعات محدود در این ابَرفریم گنجانده می‌شوند. در کل ۶ کانال کنترلی مورد استفاده قرار

می‌گیرد: پیکربندی سیستم، کنترل **بی‌درنگ** (real-time) و **غیر بی‌درنگ** (nonreal-time) ، فراخوانی (paging) ، **پاسخ دسترسی** (access response) و **پیام کوتاه** (short message) . طرز کار D-AMPS از نظر مفهومی شبیه AMPS است. وقتی گوشی تلفن همراه روشن می‌شود، با ایستگاه مرکزی تماس گرفته و بعد از معرفی خود، به کانالهای کنترلی گوش می‌کند. در این سیستم هم تعیین محل (سلول) تلفن همراه بر عهدهٔ MTSO است.

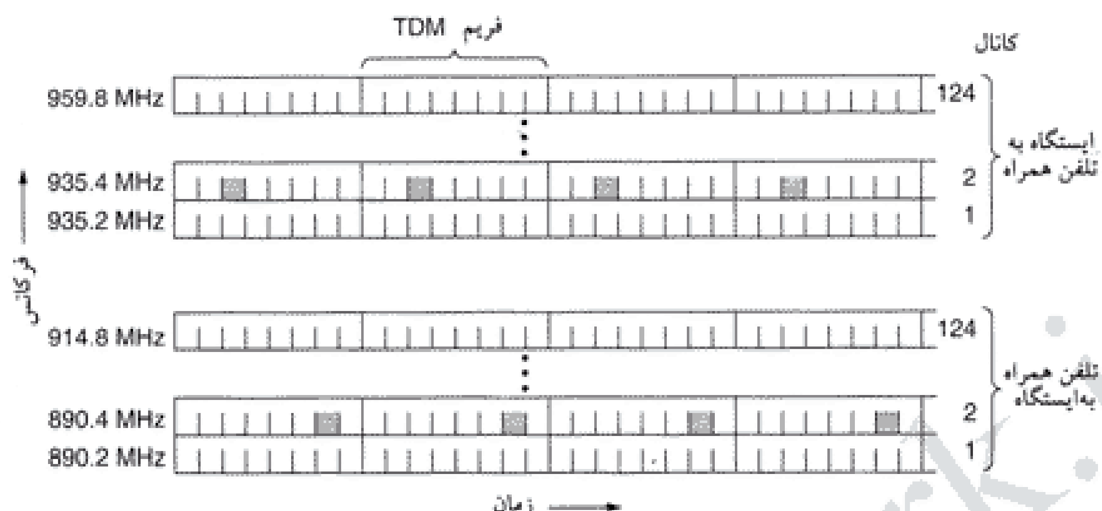
یکی از تفاوتهای D-AMPS با AMPS در نحوهٔ پاس‌کاری است. در AMPS این کار بطور کامل بر عهدهٔ MTSO است و دستگاه تلفن هیچ دخالتی در آن ندارد. اما همانطور که در شکل ۲-۴۲ دیده می‌شود، در D-AMPS در $\frac{1}{3}$ از کل زمان تماس دستگاه تلفن هیچ کاری انجام نمی‌دهد، و می‌تواند از این زمان خالی برای اندازه‌گیری کیفیت خط استفاده کند. وقتی تلفن متوجه می‌شود که قدرت سیگنال در حال کاهش است، به MTSO اطلاع داده و MTSO هم ارتباط را قطع می‌کند، که در این لحظه تلفن می‌تواند به یک ایستگاه (فرکانس) قویتر سوئیچ کند - به این تکنیک MAHO (**پاس‌کاری بکمک تلفن همراه** - Mobile Assisted HandOff) می‌گویند. در D-AMPS نیز (مانند AMPS) این فرآیند حدوداً 300 msec طول می‌کشد.

GSM : سیستم سراسری مخابرات همراه

سیستم D-AMPS بصورت گسترده در ایالات متحده (و بصورت اصلاح شده در ژاپن) در حال استفاده است. اما تقریباً در تمام نقاط دیگر دنیا از سیستمی بنام GSM (**سیستم سراسری مخابرات همراه** - Global System for Mobile Communications) استفاده می‌کنند، و حتی در آمریکا نیز این سیستم در نقاط محدودی راه‌اندازی شده است. GSM و D-AMPS از چند نظر شبیه یکدیگرند: هر دوی آنها سیستمهای سلولی هستند؛ هر دوی آنها از FDM و فرکانسهای متفاوت برای ارسال و دریافت (که فرکانس دریافت تلفن بیشتر از فرکانس ارسال آن است - 80 MHz بالاتر در D-AMPS و 55 MHz بالاتر در GSM) استفاده می‌کنند؛ و در هر دو سیستم چندین تلفن با استفاده از TDM مشترکاً از یک زوج فرکانس استفاده می‌کنند. اما در GSM پهنای کانالها بسیار بیشتر از D-AMPS (200 kHz در مقابل 30 kHz) و تعداد کاربران هر کانال نیز نسبتاً کمتر است (8 کاربر در مقابل 3 کاربر)، که در نتیجه نرخ دادهٔ آن بسیار بهتر از D-AMPS خواهد بود. در زیر توضیح مختصری دربارهٔ ویژگیهای اصلی GSM خواهیم داد. اما توجه داشته باشید که استاندارد چاپی GSM متجاوز از ۵۰۰۰ برگ است، و در آن جنبه‌های فنی و مهندسی این سیستم به تفصیل تشریح شده، که ما در اینجا بدآنهانخواهیم پرداخت.

همانطور که قبلاً هم گفته شد (و در شکل ۲-۴۳ می‌بینید)، هر باند فرکانسی 200 kHz پهنا دارد. هر سیستم GSM دارای ۱۲۴ زوج کانال **یکطرفهٔ ساده** (simplex) است، که هر کانال 200 kHz پهنا دارد و از ۸ ارتباط همزمان (بصورت مالتی‌پلکس تقسیم زمانی) پشتیبانی می‌کند. به هر ایستگاه فعال در هر بُرش زمانی یک زوج کانال تخصیص داده می‌شود، که بدین ترتیب تعداد کانالهای موجود به 992 می‌رسد. البته (بدلیل جلوگیری از تداخل فرکانسی ایستگاههای مجاور) این تعداد کانال برای تمام ایستگاهها قابل استفاده نیست. در شکل ۲-۴۳، هشت بُرش زمانی خاکستری می‌بینید که همگی متعلق به یک تماس هستند (۴ تا برای ارسال، و ۴ تا برای دریافت). ارسال و دریافت همزمان انجام نمی‌شود، چون دستگاههای رادیویی GSM برای سوئیچ کردن فرکانس به زمان نیاز دارند، و نمی‌توانند همزمان هر دو کار را انجام دهند. اگر به یک ایستگاه فرکانس 890.4/935.4 MHz اختصاص داده شده باشد و بُرش زمانی 2 بخواهد چیزی به این ایستگاه بفرستد، باید از بُرشهای خاکستری رنگ قسمت پائین تصویر (به هر تعداد که نیاز دارد) برای کار خود استفاده کند.

بُرشهای TDM نشان داده شده در شکل ۲-۴۳ بخشی از یک سلسله مراتب پیچیدهٔ فریم‌بندی هستند. هر بُرش TDM دارای ساختار خاصیست که ترکیب آنها نیز به روشی خاص تشکیل یک فریم چندگانه می‌دهد. در شکل ۲-۴۴ شکل ساده شده‌ای از این ساختار سلسله مراتبی را ملاحظه می‌کنید. همانطور که در این شکل



شکل ۲-۴۳. GSM از ۱۲۴ کانال فرکانسی استفاده می‌کند، که هر یک از آنها به ۸ بُرش زمانی تقسیم می‌شود.

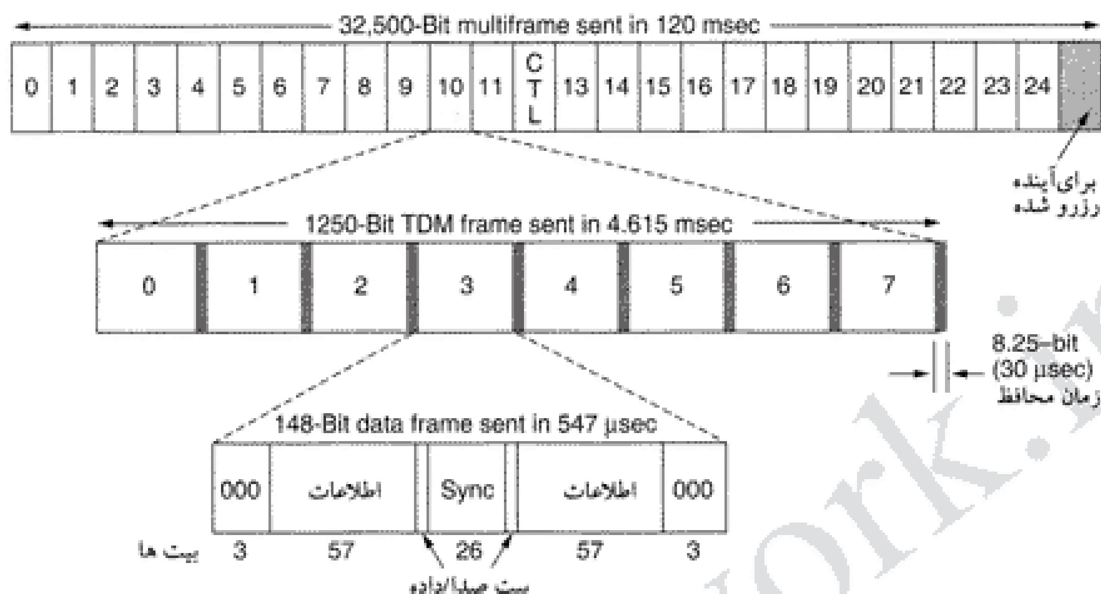
می‌بینید، هر بُرش TDM از یک فریم دادهٔ ۱۴۸ بیتی تشکیل می‌شود، که کانال را برای مدت 577 μsec (شامل 30-μsec زمان حفاظت در هر بُرش) به اشغال خود در می‌آورد. بمنظور همزمان کردن فریمها، هر فریم داده با ۳ بیت 0 شروع و ختم می‌شود. هر فریم دو فیلد اطلاعاتی (Information) ۵۷بیتی دارد، که هر کدام دارای ۱ بیت کنترلی هستند که مشخص می‌کند این فیلد اطلاعاتی حاوی داده است یا صدا. بین این فیلدهای اطلاعاتی یک فیلد نظم‌دهنده (Sync) ۲۶ بیتی وجود دارد که گیرنده از آن برای همزمان شدن با فرستنده استفاده می‌کند.

ارسال هر فریم داده 547μsec طول می‌کشد، ولی از آنجائیکه هر کانال بین ۸ ایستگاه به اشتراک گذاشته شده، فرستنده فقط در هر 4.615 msec می‌تواند یک فریم بفرستد. نرخ دادهٔ (ناخالص) هر کانال 270,833 bps است که ۸ کاربر مشترکاً از آن استفاده می‌کنند. با یک تقسیم ساده مشخص می‌شود که نرخ دادهٔ ناخالص هر کاربر 33.854 kbps (بیش از دو برابر سیستمهای D-AMPS ، یعنی 16.2 kbps) است. اما در اینجا هم مانند AMPS سرآیندها بخش زیادی از پهنای باند را می‌بلعند، و در نهایت 24.7 kbps برای دادهٔ واقعی کاربر (قبل از تصحیح خطا) باقی می‌ماند. آنچه پس از تصحیح خطا برای صدا باقی می‌ماند، 13 kbps است که باز هم کیفیت صدای بسیار بهتری نسبت به D-AMPS بدست می‌دهد (البته به قیمت مصرف پهنای باند بسیار بیشتر).

همانطور که در شکل ۲-۴۴می‌بینید، هر ۸ فریم تشکیل یک فریم TDM و هر ۲۶ فریم TDM تشکیل یک فریم چندگانهٔ 120-msec را می‌دهند. از ۲۶ فریم TDM ، فریم 12 برای کارهای کنترلی مورد استفاده قرار می‌گیرد، فریم 25 نیز برای مصارف آتی کنار گذاشته شده است، و فقط ۲۴ فریم برای ارسال و دریافت باقی می‌ماند.

علاوه بر ساختار ۲۶ فریمی شکل ۲-۴۴، در GSM از ساختار دیگری با ۵۱ فریم نیز استفاده می‌شود. در این ساختار نیز برخی از فریمها کارکرد کنترلی (مدیریت سیستم) دارند. **کانال کنترل پخش** (broadcast control channel) یکی از این کانالهاست، که استریم خروجی پیوسته‌ایست که از ایستگاه مرکزی منتشر می‌شود و شامل هویت ایستگاه و اطلاعاتی درباره وضعیت کانالهای آن می‌شود. تمام دستگاههای تلفن همراه دائماً قدرت این سیگنال را چک می‌کنند تا بتوانند موقعیت خود را در میان سلولها تشخیص دهند.

کانال کنترل اختصاصی (dedicated control channel) یکی دیگر از کانالهای کنترلی است که برای به روز در آوردن موقعیت تلفنهای همراه، ثبت آنها و برقراری تماس بکار می‌رود. بویژه، هر ایستگاه مرکزی دارای پایگاه



شکل ۲-۴۴. بخشی از ساختار فریم‌بندی GSM .

داده‌ای از تمام تلفنهای موجود در محدودهٔ قدرت خود است، که با استفاده از اطلاعاتی که روی این کانال فرستاده می‌شود، به روز در می‌آید.

دیگر کانال کنترلی، **کانال کنترل مشترک** (common control channel) است که به سه زیرکانال تقسیم می‌شود. اولین زیرکانال، **کانال فراخوانی** (paging channel) است که ایستگاه مرکزی از آن برای اعلام تماس ورودی استفاده می‌کند (و هر تلفن دائماً این کانال را چک می‌کند تا ببیند آیا تماسی دارد یا خیر). زیرکانال دوم، **کانال دسترسی تصادفی** (random access channel) است، که اجازه می‌دهد تا کاربران درخواستی برای یک بُرش از کانال کنترلی اختصاصی را روی آن ارسال کنند؛ کاربران از این بُرش برای برقراری تماس با دیگران استفاده می‌کنند. اگر دو کاربر همزمان وارد این کانال شوند، تداخل پیش می‌آید و باید (پس از کمی تأخیر) عملیات را از نو تکرار کنند. بُرش اختصاص داده شده به کاربر روی زیرکانال سوم، **کانال اختصاص دسترسی** (access grant channel) ، به وی اعلام می‌شود.

CDMA : دسترسی چندگانه با تقسیم کُد

سیستمهای D-AMPS و GSM هر دو سیستمهایی نسبتاً سنتی و معمولی هستند، که از FDM و TDM برای تقسیم طیف فرکانسی به کانالها و تقسیم کانالها به بُرشهای زمانی استفاده می‌کنند. اما بازیگر سومی بنام CDMA (**دسترسی چندگانه با تقسیم کُد** – Code Division Multiple Access) نیز در این صحنه حاضر است، که به روشی کاملاً متفاوت بازی می‌کند. وقتی CDMA برای اولین بار در صنعت مطرح شد، همان واکنشی را برانگیخت که پیشنهاد کریستف کلمب برای رسیدن به هندوستان از راه سفر به غرب. اما در نتیجهٔ پایداری و مقاومت یک شرکت بنام Qualcomm ، اکنون CDMA به جایی رسیده که نه تنها بعنوان یک سیستم قابل‌قبول مطرح است، بلکه به آن به چشم تنها مبنای مطمئن برای سیستمهای تلفن همراه نسل سوم نگاه می‌کنند. در ایالات متحده، CDMA حتی اکنون (در سیستمهای نسل دوم) نیز بعنوان رقیبی جدی برای D-AMPS مطرح است – برای مثال، شرکت Sprint PCS از CDMA استفاده می‌کند، در حالیکه AT&T Wireless از D-AMPS . استاندارد CDMA در سند IS-95 تدوین شده، و گاهی به این نام هم شناخته می‌شود؛ نام cdmaOne نیز یکی از نامهای تجاری آن است.

CDMA بکلی از AMPS ، D-AMPS و GSM متفاوت است: بجای تقسیم طیف فرکانسی به کانالهای باریک، CDMA اجازه می‌دهد تا تمام تلفنها و ایستگاهها از تمام طیف فرکانسی برای ارسال و دریافت استفاده کنند، و برای تفکیک آنها از یکدیگر از **تئوری رمزگذاری** (*coding theory*) استفاده می‌کند. در سیستمهای CDMA این فرض که فریمهای تداخل کرده غیر قابل استفاده‌اند را نیز بکلی کنار می‌گذارد، و بجای آن فرض می‌کند که این سیگنالها بصورت خطی با هم جمع می‌شوند.

برای درک بهتر CDMA از یک مثال آشنا استفاده می‌کنیم: سالنی پر از جمعیت که دو به دو مشغول صحبت هستند. *TDM* مانند آن است که این افراد را وسط سالن جمع کنید، ولی فقط به نوبت به آنها اجازهٔ صحبت کردن بدهید. FDM مانند آن است که این افراد را با فاصلهٔ زیاد از یکدیگر بچینید، و به آنها اجازه دهید دو به دو همزمان (و البته مستقل) با هم صحبت کنند. CDMA نیز مانند این است که همهٔ افراد را وسط سالن جمع کنید، ولی آنها (همزمان) به زبانهای مختلف با هم صحبت کنند برای مثال، دو نفری که به فرانسه با هم حرف می‌زنند، هر چیزی غیر از کلمات فرانسوی را بعنوان پارازیت (نویز) نشنیده می‌گیرند. نکتهٔ کلیدی در CDMA همین استخراج سیگنال موردنظر (و دور ریختن هر چیز دیگر) است. در زیر روش کار یک سیستم ساده شدهٔ CDMA را توضیح می‌دهیم.

در CDMA ، هر بیت به m فاصلهٔ زمانی کوتاه، موسوم به **چیپ** (chip) ، تقسیم می‌شود معمولاً هر بیت دارای ۶۴ یا ۱۲۸ چیپ است، ولی در اینجا برای سادگی هر بیت را به فقط ۸ چیپ تقسیم کرده‌ایم.

به هر تلفن همراه (یا ایستگاه) یک کُد m بیتی منحصربفرد، که به آن **توالی چیپ** (chip sequence) می‌گویند، اختصاص داده می‌شود برای ارسال بیت 1 ، ایستگاه توالی چیپ خود را می‌فرستد، و برای ارسال بیت 0 **مکمل یک** (one's complement) توالی چیپ خود را - یک ایستگاه مجاز نیست هیچ چیز دیگری بفرستد. برای مثال، با فرض $m = 8$ ، اگر ایستگاهی بنام A دارای توالی چیپ 00011011 باشد، برای ارسال بیت 1 توالی 00011011 و برای ارسال 0 توالی 11100100 را می‌فرستد.

افزایش مقدار اطلاعات ارسالی از b bits/sec به mb chips/sec فقط وقتی امکانپذیر است که پهنای باند موجود m برابر شود، که بدین‌ترتیب CDMA به سیستمی با طیف گسترده تبدیل می‌شود. اگر پهنای باند موجود برای ۱۰۰ ایستگاه 1 MHz باشد، با تکنیکهای FDM هر ایستگاه فقط 10 kHz در اختیار خواهد داشت و می‌تواند حداکثر 10 kbps (با فرض 1 bit/Hz) ارسال کند. با CDMA هر ایستگاه می‌تواند از تمامی پهنای باند 1-Mhz استفاده کند، و به سرعت 1 megachips/sec برسد. اگر تعداد **چیپ بر بیت** کمتر از ۱۰۰ باشد، پهنای باند مؤثر CDMA باز هم بیشتر از FDM است (و مشکلات **تخصیص کانال** نیز دیگر وجود ندارد).

در این قسمت برای درک بهتر مطلب روش **دو-علامتی** را بکار می‌بریم، یعنی بجای 0 باینری از 1- و بجای 1 باینری از 1+ استفاده خواهیم کرد. توالی چیپ‌ها را نیز در پرانتز نمایش می‌دهیم، که بدین‌ترتیب توالی چیپ ایستگاه A به (1+ 1+ 1- 1+ 1+ 1- 1- 1-) تبدیل می‌شود. در شکل ۲-۴۵ (الف) توالی چیپ چهار ایستگاه بنامهای A ، B ، C و D را می‌بینید؛ در شکل ۲-۴۵ (ب) نیز همین توالی‌ها را به روش دو-علامتی نشان داده‌ایم.

هر ایستگاه توالی چیپ خاص خود را دارد. فرض می‌کنیم **S** بردار mچیپی ایستگاه S ، و $\bar{\mathbf{S}}$ بردار منفی (مکمل یک) آن است. تمام بردارهای توالی چیپ **متعامد** (orthogonal) هستند، بعبارت دیگر ضرب داخلی نرمال شدهٔ هر دو بردار **S** و **T** ($\mathbf{S} \bullet \mathbf{T}$) صفر است. با استفاده از روشی بنام **کُدهای والش** (Walsh codes) می‌توان چنین بردارهایی تولید کرد. متعامد بودن دو بردار را به زبان ریاضی می‌توان چنین نوشت:

$$\mathbf{S} \bullet \mathbf{T} \equiv \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0$$

همانطور که خواهید دید، متعامد بودن بردارها یکی از ویژگیهای کلیدی این سیستم است. توجه داشته باشید که اگر $\mathbf{S} \bullet \mathbf{T} = 0$، آنگاه $\mathbf{S} \bullet \overline{\mathbf{T}} = 0$. ضرب داخلی نرمال شدهٔ هر بردار در خودش نیز 1 است:

$$\mathbf{S} \bullet \mathbf{S} = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1$$

از آنجائیکه هر یک از جملات این دنباله 1 است، مجموع m جملهٔ آن معادل m می‌شود، که بعد از ضرب در $\frac{1}{m}$ به عدد 1 خواهیم رسید. همچنین توجه داشته باشید که، $\mathbf{S} \bullet \overline{\mathbf{S}} = -1$.

همانطور که گفتیم، هر ایستگاه می‌تواند برای ارسال بیت 1 توالی چیپ خود، و برای ارسال بیت 0 مکمل یک توالی چیپ خود را بفرستد، و یا اصلاً سکوت کند و چیزی نفرستد. فعلاً فرض را بر این می‌گذاریم که تمام ایستگاهها سنکرون هستند، یعنی ارسال توالی‌های خود را در یک زمان شروع می‌کنند.

وقتی دو یا چند ایستگاه بطور همزمان شروع به ارسال می‌کنند، سیگنال دو-علامتی آنها بصورت خطی با هم جمع می‌شود. برای مثال، اگر (در دورهٔ زمانی یک چیپ) سه ایستگاه سیگنال 1+ و یک ایستگاه سیگنال 1- بفرستند، مجموع آنها 2+ خواهد شد. می‌توان این وضعیت را مانند جمع ولتاژها تصور کرد: مجموع سه ولتاژ 1+ ولت و یک ولتاژ 1- ولت معادل 2+ ولت می‌شود. در شکل ۲-۴۵(ج) شش نمونه از ارسال همزمان یک یا چند ایستگاه را ملاحظه می‌کنید. در مثال اول، ایستگاه C مبادرت به ارسال یک بیت 1 می‌کند، بعبارت دیگر توالی چیپ خود (01011100) را می‌فرستد. در مثال دوم، هر دو ایستگاه B و C یک بیت 1 می‌فرستند، که بدین ترتیب مجموع توالی‌های دو-علامتی آنها چنین خواهد شد:

(-1 -1 +1 -1 +1 +1 +1 -1) + (-1 +1 -1 +1 +1 +1 -1 -1) = (-2 0 0 0 +2 +2 0 -2)

(الف)	(ب)
A: 0 0 0 1 1 0 1 1	A: (−1 −1 −1 +1 +1 −1 +1 +1)
B: 0 0 1 0 1 1 1 0	B: (−1 −1 +1 −1 +1 +1 +1 −1)
C: 0 1 0 1 1 1 0 0	C: (−1 +1 −1 +1 +1 +1 −1 −1)
D: 0 1 0 0 0 0 1 0	D: (−1 +1 −1 −1 −1 −1 +1 −1)

شش مثال:

– – 1 –	C	S_1 = (−1 +1 −1 +1 +1 +1 −1 −1)
– 1 1 –	B + C	S_2 = (−2 0 0 0 +2 +2 0 −2)
1 0 – –	A + $\overline{B}$	S_3 = (0 0 −2 +2 0 −2 0 +2)
1 0 1 –	A + $\overline{B}$ + C	S_4 = (−1 +1 −3 +3 +1 −1 −1 +1)
1 1 1 1	A + B + C + D	S_5 = (−4 0 −2 0 +2 0 +2 −2)
1 1 0 1	A + B + $\overline{C}$ + D	S_6 = (−2 −2 0 −2 0 −2 +4 0)

(ج)

$S_1 \bullet C$ = (1 +1 +1 +1 +1 +1 +1 +1)/8 = 1
$S_2 \bullet C$ = (2 +0 +0 +0 +2 +2 +0 +2)/8 = 1
$S_3 \bullet C$ = (0 +0 +2 +2 +0 −2 +0 −2)/8 = 0
$S_4 \bullet C$ = (1 +1 +3 +3 +1 −1 +1 −1)/8 = 1
$S_5 \bullet C$ = (4 +0 +2 +0 +2 +0 −2 +2)/8 = 1
$S_6 \bullet C$ = (2 −2 +0 −2 +0 −2 −4 +0)/8 = −1

(د)

شکل ۲-۴۵. (الف) توالی چیپ باینری چهار ایستگاه. (ب) توالی چیپ دو-علامتی همان ایستگاهها. (ج) شش نمونه از ارسال همزمان ایستگاهها. (د) استخراج سیگنال ایستگاه C .

در مثال سوم، ایستگاه A یک بیت 1 و ایستگاه B یک بیت 0 می‌فرستد، و بقیهٔ ایستگاهها ساکت هستند. در مثال چهارم، ایستگاههای A و C یک بیت 1 و ایستگاه B یک بیت 0 می‌فرستند. در مثال پنجم، هر چهار ایستگاه یک بیت 1 می‌فرستند؛ و بالاخره در مثال آخر، ایستگاههای A ، B و D یک بیت 1 می‌فرستند و ایستگاه C یک بیت 0 . توجه داشته باشید که در تمام مثالهای فوق، توالی‌های S_1 تا S_6 فقط یک بیت را نشان می‌دهند.

برای تشخیص و استخراج استریم بیت‌های یک ایستگاه باید توالی چیپ آن ایستگاه را از قبل بدانیم. این کار را می‌توان با محاسبهٔ ضرب داخلی نرمال شدهٔ توالی چیپ دریافت شده (جمع خطی سیگنال تمام ایستگاهها در آن لحظه) با توالی چیپ ایستگاه موردنظر انجام داد. اگر توالی دریافت شده را **S** بنامیم، و بخواهیم توالی چیپ (سیگنال ارسالی) ایستگاه **C** را از آن بیرون بکشیم، کافیست ضرب داخلی نرمال شدهٔ $\mathbf{S \bullet C}$ را محاسبه کنیم.

برای اینکه ببینید این روش چگونه کار می‌کند، مثال چهارم شکل ۲-۴۵ (ج) را در نظر بگیرید. چیزی که گیرنده دریافت می‌کند، $\mathbf{S = A + \overline{B} + C}$ است، و برای استخراج سیگنال **C** بایستی عبارت $\mathbf{S \bullet C}$ را محاسبه کند:

$$\mathbf{S \bullet C = (A + \overline{B} + C) \bullet C = A \bullet C + \overline{B} \bullet C + C \bullet C} = 0 + 0 + 1 = 1$$

همانطور که می‌بینید، دو جملهٔ اول (بدلیل متعامد بودن بردارها) صفر شده‌اند. از اینجا می‌توانید علت انتخاب بردارهای متعامد را دریابید.

این وضعیت را می‌توان بصورت سه سیگنال مستقل نیز در نظر گرفت: سیگنال‌ها بصورت مستقل و جدا از هم دریافت شده، و پس از محاسبهٔ ضرب داخلی، با هم جمع می‌شوند. بعلت متعامد بودن بردارها، تمام ضربهای داخلی (بجز $\mathbf{C \bullet C}$) صفر خواهند شد. بعبارت دیگر، تقدم ضرب داخلی یا جمع تأثیری بر نتیجهٔ نهایی ندارد.

برای درک بهتر روش از رمز خارج کردن سیگنالها به شکل ۲-۴۵ (د) نگاه کنید. فرض کنید گیرنده می‌خواهد از شش سیگنال S_1 تا S_6 بیت‌های ارسال شده از ایستگاه C را استخراج کند. برای این کار، گیرنده تک تک سیگنالهای **S** را در بردار **C** (شکل ۲-۴۵ ب) ضرب کرده، و سپس 1/8 آنرا محاسبه می‌کند (چون، $m = 8$). همانطور که می‌بینید، گیرنده توانسته است بطور صحیح بیتهای ارسالی از C را استخراج کند.

در یک سیستم ایده‌آل (بدون نویز) CDMA تعداد ایستگاهها را می‌توان به تعداد دلخواه زیاد کرد (همانگونه که در یک کانال بدون نویز نایکوئیست می‌توان نرخ نمونه‌برداری را بدلخواه افزایش داد). اما در عمل، محدودیتهای فیزیکی ظرفیت سیستم را بشدت پائین می‌آورند. اول اینکه، فرض کردیم تمام چیپ‌ها از نظر زمانی سنکرون هستند، در حالیکه این وضعیت عملاً غیرممکن است. بهترین کاری که می‌توان کرد اینست که فرستنده با ارسال یک توالی چیپ از پیش تعریف شده (که باندازهٔ کافی طولانی است) به گیرنده امکان دهد تا خود را با فرستنده سنکرون کند. در این حالت تمام سیگنالهای غیرسنکرون بعنوان نویز تلقی خواهند شد. اگر تعداد این سیگنالهای غیرسنکرون چندان زیاد نباشد، الگوریتم فوق همچنان بخوبی کار خواهد کرد. در زمینهٔ **بر هم نهی** (superposition) توالی‌های چیپ با سطح نویز تحقیقات تئوریک مفصلی انجام شده است (Pickholtz et al., 1982) . همانطور که می‌توان انتظار داشت، هر چه توالی چیپ طولانی‌تر باشد، احتمال استخراج آن در محیط‌های پُرنویز بیشتر خواهد بود. حتی می‌توان در توالی بیت‌ها از کُدهای تصحیح خطا نیز استفاده کرد - البته در توالی چیپ هرگز از کُدهای تصحیح خطا استفاده نمی‌شود.

یکی دیگر از مفروضات ضمنی بحث فوق یکسان بودن قدرت سیگنالهائیست که به گیرنده می‌رسند. در سیستمهای تلفن همراه (و از جمله CDMA) فاصلهٔ تلفنهای همراه از ایستگاه مرکزی (و در نتیجه قدرت تشعشعی آنها) متغیر است، و قدرت سیگنالهایی که از تلفنها به ایستگاه مرکزی می‌رسد یکسان نیست. یکی از روشهای ابتکاری برای حل این مشکل آنست که تلفنهای همراه با کاهش سطح سیگنال دریافتی از ایستگاه مرکزی، توان تشعشعی خود را بالا ببرند؛ بعبارت دیگر، هر چه از ایستگاه مرکزی دور می‌شوند، سیگنال قویتری بفرستند.

ایستگاه مرکزی نیز می‌تواند صریحاً به تلفنهای همراه فرمان دهد تا (بسته به فاصله‌شان از ایستگاه) قدرت سیگنالهای خود را افزایش یا کاهش دهند.

همچنین فرض کردیم که گیرنده از هویت فرستنده آگاه است. از نظر تئوری این امکان هست که (با فرض وجود قدرت محاسباتی)، یک گیرنده به تمام سیگنالها گوش دهد و الگوریتمهای از رمز خارج کردن را روی تمام آنها اجرا کند. اما مثل معروفی هست که می‌گوید، «حرف زدن همیشه از عمل کردن ساده‌تر است.» CDMA پیچیدگیهای دیگری نیز دارد، که در این بحث مختصر آنها را نادیده گرفتیم. با این حال، CDMA یکی از هوشمندانه‌ترین سیستمهای مخابرات بیسیم محسوب می‌شود، که بسرعت در حال گسترش است. این سیستم معمولاً از یک باند 1.25 MHz (برخلاف 30 kHz در D-AMPS ، و 200 kHz در GSM) استفاده می‌کند، و تعداد کاربرانی که پشتیبانی می‌کند، از هر دو سیستم قبلی بیشتر است. در عمل، پهنای باند موجود برای هر کاربر در سیستمهای CDMA حتی از GSM هم بهتر است.

یکی از بهترین منابع موجود در این زمینه (Lee and Miller, 1998) است. روشهای دیگری نیز در (Crespo et al., 1995) و (Sari et al., 2000) تشریح شده‌اند، که البته فهم آنها به دانش زیادی در زمینهٔ مهندسی مخابرات نیاز دارد.

۲-۶-۳　تلفن‌های همراه نسل سوم: صدای دیجیتال و داده

آیندهٔ تلفن همراه چیست؟ اجازه دهید نگاه سریعی به آن بیندازیم. صنعت تلفن همراه عوامل محرک و پیشبرندهٔ متعددی دارد. اول، ترافیک داده مدتهاست از ترافیک صدا پیشی گرفته و همچنان به رشد تصاعدی خود ادامه می‌دهد، ولی ترافیک صدا سالهاست دیگر رشد چندانی را تجربه نمی‌کند. بسیاری از متخصصان پیش‌بینی می‌کنند که بزودی در تلفنهای همراه نیز شاهد پیشی گرفتن ترافیک داده از ترافیک صدا خواهیم بود. دوم، صنایع تلفن، تفریحات و کامپیوتر همگی دیجیتالی شده‌اند و روز بروز بیشتر به یکدیگر نزدیک می‌شوند. مردم مدتهاست با شور و هیجان از دستگاه کوچک، سبک و قابل‌حملی صحبت می‌کنند که بتواند بعنوان تلفن، پخش CD ، پخش DVD ، ترمینال ایمیل و وب، وسیلهٔ بازی و سرگرمی، واژه‌پرداز (و غیره و غیره) کار کند، و قادر باشد در هر نقطه‌ای بصورت بیسیم و با پهنای باند بالا به اینترنت متصل شود. این دستگاه رؤیایی همان تلفن همراه نسل سوم است؛ برای اطلاعات بیشتر (Huber et al., 2000; and Sarikaya, 2000) را ببینید.

در سال ۱۹۹۲، ITU کوشید این رؤیا را کمی بیشتر به واقعیت نزدیک کند، و به همین منظور طرح اولیه‌ای بنام IMT-2000 (که IMT مخفف سیستم مخابرات تلفن همراه بین‌المللی - International Mobile Telecommunication - است) منتشر کرد. عدد 2000 سه چیز را نشان می‌داد: ۱) سالی که این سیستم بایستی عملیاتی شود، ۲) فرکانسی (بر حسب MHz) که سیستم تحت آن کار می‌کند، و ۳) پهنای باند این سرویس (بر حسب kHz).

البته IMT-2000 به هیچیک از این اهداف نرسید. در سال ۲۰۰۰ هیچ سیستمی نصب و راه‌اندازی نشد. ITU پیشنهاد کرده بود که دولتها طیف 2 GHz را برای این منظور کنار بگذارند تا این سرویس بتواند بین کشورهای مختلف بدون اشکال کار کند؛ تنها کشوری که به این توصیه عمل کرد، چین بود. و بالاخره، مشخص شد که در حال حاضر پهنای باند 2 Mbps برای کاربرانی که بیش از حد متحرک هستند، عملی نیست (علت آن هم دشواری پاس‌کاری این قبیل کاربران است). پهنای باند عملی‌تر عبارتست از 2 Mbps برای کاربران ثابت خانگی (که می‌تواند با ADSL رقابت کند)، 384 kpbs برای کاربرانی که قدم می‌زنند، و 144 kbps برای آنهایی که سوار اتومبیل هستند. با این حال، حوزهٔ فعالیت 3G (نسل سوم) بسیار پُر جنب و جوش است. نسل سوم شاید کمی دیرتر از راه برسد و کمی کمتر از آنچه که انتظار می‌رود باشد، ولی حتماً می‌آید.

سرویسهایی که قرار است شبکهٔ IMT-2000 در اختیار کاربران خود بگذارد، عبارتند از:

۱. انتقال صدا با کیفیت عالی.
۲. پیام‌رسانی (سرویسی که جایگزین ایمیل، فکس، SMS ، chat و غیره خواهد شد).
۳. مالتی‌مدیا (پخش موسیقی، تماشای ویدئو، فیلم، تلویزیون و غیره).
۴. دسترسی اینترنت (گشت و گذار در وب، از جمله صفحاتی که صدا، تصویر و فیلم دارند)

سرویسهای دیگری از قبیل **کنفرانس ویدئویی** (video conferencing) ، **حضور از راه دور** (telepresence) ، بازیهای گروهی و **تجارت-همراه** (m-commerce : پرداخت بهای اجناس خریداری شده در فروشگاه با نزدیک کردن تلفن همراه به صندوق) را نیز می‌توان از این شبکه انتظار داشت. علاوه بر آن، تمام این سرویسها بین‌المللی خواهند بود (یعنی در تمام نقاط دنیا می‌توان به آنها دسترسی داشت: در جاهایی که خطوط ارتباطی زمینی وجود ندارد، تلفن همراه بطور خودکار از لینکهای ماهواره‌ای استفاده خواهد کرد)، و همیشه با کیفیت تضمین شده در دسترس هستند.

ITU برای IMT-2000 تکنولوژی واحدی را در نظر گرفته است، بگونه‌ایکه تلفنهای همراه بتوانند در هر نقطه‌ای از دنیا کار کنند (مانند دستگاههای ضبط صوت و پخش CD ، نه تلویزیونها و تلفنهای همراه امروزی). یکسان شدن تکنولوژی، علاوه بر تسهیل کار شرکتهای مخابرات، افراد بیشتری را به استفاده از این سرویسها تشویق خواهد کرد. جنگ تکنولوژیها (مانند آنچه بین ویدئوهای Betamax و VHS رخ داد) هیچگاه به نفع صنعت و تجارت نبوده است.

پیشنهادهای متعددی ارائه شد، که بعد از غربال شدن آنها، سرانجام دو طرح باقی ماند. طرح اول، بنام W-CDMA (CDMA **پهن‌باند** – Wideband CDMA)، از طرف شرکت سوئدی اریکسون ارائه شد. این طرح از توالی مستقیم با طیف گسترده (مانند آنچه در بالا گفتیم) استفاده می‌کند. این سیستم در یک باند 5-MHz کار می‌کند، و بگونه‌ای طراحی شده که بتواند با شبکه‌های GSM (البته نه GSM قدیمی) کار کند. تلفنهای این سیستم می‌توانند بدون اختلال در ارتباط یک سلول W-CDMA را ترک کرده و وارد یک سلول GSM شوند. اتحادیهٔ اروپا بشدت از این سیستم پشتیبانی می‌کند، و به آن نام UMTS (**سیستم مخابرات تلفن همراه جهانی** – Universal Mobile Telecommunications System) را داده است.

طرح دیگر که از طرف شرکت Qualcomm (مخترع CDMA) پیشنهاد شده، CDMA2000 نام دارد. این سیستم نیز اساساً همان توالی مستقیم با طیف گسترده (شکل اصلاح شدهٔ استاندارد IS-95) است، که با آن سازگاری دارد. CDMA2000 نیز از یک باند 5-MHz استفاده می‌کند، ولی نمی‌تواند با سلولهای GSM (و طبیعتاً، با سلولهای D-AMPS) پاس‌کاری انجام دهد. CDMA2000 تفاوتهای دیگری، از قبیل نرخ چیپ، زمان فریم، طیف فرکانسی، و روش سنکرون شدن، نیز با W-CDMA دارد.

اگر مهندسان اریکسون و Qualcomm را در اتاقی حبس کنند و از آنها بخواهند یک سیستم مشترک طراحی کنند، به احتمال زیاد می‌توانند. بهر حال، هر دو سیستم از تکنیکهای CDMA و یک کانال 5-MHz استفاده می‌کنند، و چیزهای دیگر هم مسلماً ارزش خودکشی ندارند. در اینجا هم (مثل همیشه) مشکل نه فنی و مهندسی بلکه سیاسی است. اروپا سیستمی می‌خواهد که با GSM سازگاری داشته باشد، و ایالات متحدهٔ آمریکا دنبال سیستمی است که با IS-95 سازگار باشد. هر دو نیز از شرکتهای محلی خود دفاع می‌کنند: اروپا از اریکسون (که شرکتی سوئدی است) و آمریکا از Qualcomm (که در کالیفرنیا است). اریکسون و Qualcomm دعواهای حقوقی بیشماری نیز بر سر حق‌اختراع CDMA داشته‌اند.

بالاخره در مارس ۱۹۹۹ اریکسون Qualcomm را خرید، و دعواها خاتمه یافت. آنها بر سر یک استاندارد 3G نیز به توافق رسیدند، ولی این استاندارد ناسازگاری‌های زیادی داشت. اما این منازعات بالاخره به پایان می‌رسد، و بزودی شاهد تلفنها و سرویسهای 3G خواهیم بود.

دربارهٔ سیستمهای 3G مطالب زیادی نوشته شده، و برخی آنرا بعنوان یکی از بزرگترین اختراعات بشری ستوده‌اند. برای نمونه به (Collins and Smith, 2000; De Vriendt et al., 2002; Harte et al., 2002; 2002; Lu, and Sarikaya, 2000) اشاره می‌کنیم. اما 3G منتقدانی نیز دارد که معتقدند اساساً راه را اشتباه می‌رود، و می‌توان از میان آنها از (Garber, 2002; and Goodman, 2000) نام برد.

عده‌ای هم که از جنگ 3G خسته شده‌اند، به فکر افتادند تا قدم کوچکی به سمت آن بردارند، و نام آنرا هم 2.5G گذاشتند (اگر چه شاید نام 2.1G برای آن مناسب‌تر است). یکی از این سیستمها EDGE (**نرخ دادهٔ بهبودیافته برای تکامل** GSM - Enhanced Data rates for GSM Evolution) نام دارد، که اساساً چیزی نیست جز همان GSM با bits/baud بیشتر. مشکل اینجاست که bits/baud بیشتر یعنی خطای بیشتر، و به همین دلیل EDGE در سرعتهای متفاوت از ۹ روش مختلف برای مدولاسیون و تصحیح خطا استفاده می‌کند.

یکی دیگر از طرحهای 2.5G ، GPRS (**سرویس عمومی بستهٔ رادیویی** - General Packet Radio Service) نام دارد، که عبارتست از یک **شبکهٔ سوئیچینگ بسته‌ای** که روی GSM یا D-AMPS کار می‌کند. این سیستم اجازه می‌دهد تا تلفنهای همراه در سلولهای صوتی بسته‌های IP رد و بدل کنند. GPRS برای این منظور از بُرشهای زمانی و فرکانسهای اختصاصی استفاده می‌کند، که تعداد و محل آنها (به نسبت ترافیک صوت و داده در سلول) بطور دینامیک توسط ایستگاه مرکزی تعیین می‌شود.

بُرشهای زمانی موجود به چندین کانال منطقی تقسیم شده، و به مصارف مختلف می‌رسند. تخصیص بُرشهای زمانی به هر کانال توسط ایستگاه مرکزی صورت می‌گیرد. یکی از این کانالهای منطقی برای ارسال بسته‌ها از ایستگاه مرکزی به تلفنهای همراه است، و هر بسته می‌داند که مقصد آن کجاست. برای ارسال یک بستهٔ IP ، تلفن همراه درخواستی برای تخصیص یک یا چند بُرش زمانی به ایستگاه مرکزی می‌فرستد. اگر این درخواست بدون مشکل به ایستگاه مرکزی برسد، ایستگاه مرکزی فرکانس و بُرشهای زمانی تخصیص یافته را به تلفن همراه اعلام می‌کند. همین که بسته‌های IP ارسالی از تلفن همراه به ایستگاه مرکزی رسید، ایستگاه مرکزی (از طریق ارتباطاتی که دارد) آنها را به اینترنت منتقل می‌کند. از آنجائیکه GPRS فقط لایه‌ایست روی سیستمهای صوتی موجود، می‌توان آنرا در بهترین حالت یک قدم به سمت 3G بشمار آورد.

با اینکه شبکه‌های 3G هنوز بطور کامل راه‌اندازی نشده‌اند، برخی از محققان آنرا موضوعی تحقق‌یافته (که دیگر ارزش توجه ندارد) تلقی کرده و سراغ سیستمهای **نسل چهارم** (4G) رفته‌اند (Berezdivin et al., 2002; Guo and Chaskar, 2002; Huang and Zhuang, 2002; Kellerer et al., 2002; and Misra et al., 2002) . برخی از مشخصات پیشنهادی سیستمهای 4G عبارتند از: پهنای باند زیاد، دسترسی در همه جا، یکپارچگی کامل با شبکه‌های کابلی (و بویژه شبکه‌های IP)، مدیریت تطبیقی منابع و طیف فرکانسی، رادیوی نرم‌افزاری، و سرویسهای مالتی‌مدیا با کیفیت عالی.

از سوی دیگر، با توجه به تعداد بسیار زیاد شبکه‌های محلی بیسیم 802.11 که امروزه نصب شده، برخی معتقدند 3G به دنیا نیامده، مرده است. اینها می‌گویند، «هر جا که بروید بالاخره تحت پوشش یک شبکهٔ 802.11 هستید، و این همان چیزیست که 3G می‌خواهد با آن همه منت به شما بدهد.» پنج سال دیگر معلوم می‌شود حق با چه کسی بوده است.

۲ـ۷ تلویزیون کابلی

تا اینجا سیستمهای تلفن ثابت و بیسیم را بررسی کردیم. هر دوی این سیستمها مسلماً نقش مهمی در آیندهٔ شبکه بازی خواهند کرد. اما بازیگر دیگری نیز در زمینهٔ شبکه‌های ثابت وارد صحنه شده، و هر روز اهمیت بیشتری می‌یابد: **تلویزیون کابلی** (Cable TV) . امروزه افراد بسیاری سرویسهای تلویزیون و اینترنت خود را از طریق کابل دریافت می‌کنند، و شرکتهای تلویزیون کابلی نیز با جدیت بدنبال سهم بیشتری از بازار هستند. در این قسمت تلویزیون کابلی را از دیدگاه شبکه مورد بررسی دقیقتر قرار خواهیم داد. برای کسب اطلاعات بیشتر نیز می‌توانید به (Laubach et al., 2001; Louis, 2002; Ovadia, 2001; and Smith, 2002) مراجعه کنید.

۲ـ۷ـ۱ تلویزیون با آنتن مرکزی

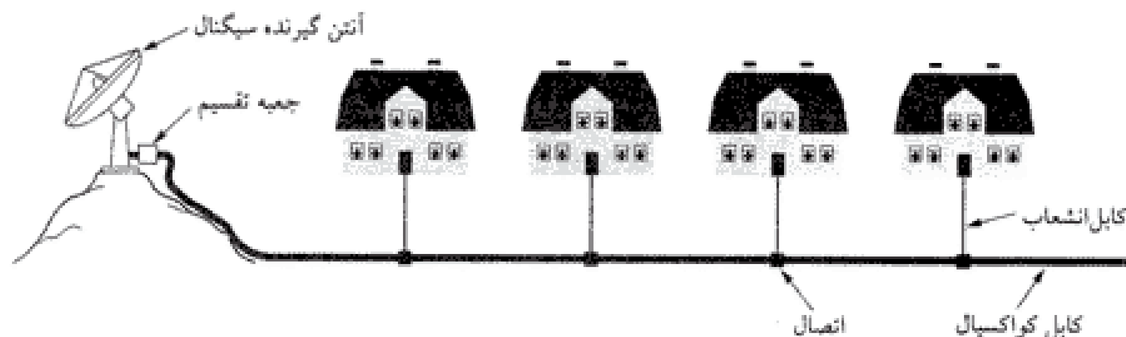
تلویزیون کابلی در اواخر دههٔ ۱۹۴۰ بعنوان راهی برای ارائهٔ سرویسهای بهتر به مناطق روستایی و کوهستانی ابداع شد. این سیستم عبارت بود از یک آنتن بزرگ بر فراز نقطه‌ای مرتفع (برای دریافت امواج تلویزیونی)، یک تقویت‌کننده (موسوم به **جعبه تقسیم** – head end) برای تقویت سیگنال، و یک رشته کابل کواکسیال برای انتقال سیگنال تلویزیون به منازل؛ شکل ۲-۴۶ را ببینید.

در آن سالها به این سیستم **تلویزیون با آنتن مرکزی** (Community Antenna Television) گفته می‌شد. این سیستم بقدری ساده بود که هر کسی با کمترین تجربهٔ سیمکشی نیز می‌توانست آنرا راه بیندازد، و هزینهٔ آن هم معمولاً بین مشترکین سرشکن می‌شد. با اضافه شدن مصرف‌کنندگان فقط کافی بود انشعابهای بیشتری از کابل اصلی گرفته (و احیاناً تقویت‌کننده‌های بیشتری نصب) شود. این سیستم اساساً یکطرفه (از جعبه تقسیم سیگنال به مشترکین) بود، و تا دههٔ ۱۹۷۰ هزاران نمونه از آن نصب شده بود.

در سال ۱۹۷۴ شرکت رسانه‌ای تایم کانال جدیدی بنام Home Box Office راه‌اندازی کرد که فیلمهای سینمایی پخش می‌کرد، و فقط از طریق کابل قابل دسترسی بود. کانالهای کابلی دیگر نیز بسرعت راه افتادند که اخبار، مسابقات ورزشی، آشپزی (و بسیاری موضوعات دیگر) پخش می‌کردند. این تحول باعث دو تغییر اساسی در صنعت تلویزیون شد. اول اینکه، شرکتهای بزرگ شروع به خریدن شبکه‌های کابلی موجود در شهرها کردند، و در بسیاری جاها نیز خود رأساً شروع به کابل‌کشی و جذب مشترکین جدید کردند. دوم اینکه، برای ارائهٔ سرویس در سراسر کشور لازم بود بین شهرها نیز کابل‌کشی شود، پس شرکتهای مزبور شروع به متصل کردن شهرها به یکدیگر کردند. این وضعیت درست مانند شبکهٔ تلفن در ۸۰ سال قبل بود، که شرکتهای تلفن برای ارائهٔ سرویسهای راه دور بین شهرها کابل تلفن کشیدند.

۲ـ۷ـ۲ اینترنت کابلی

در طول سالها سیستمهای کابلی رشد کرد، و بین شهرهای مختلف فیبرهای نوری با پهنای باند زیاد کشیده شد. این

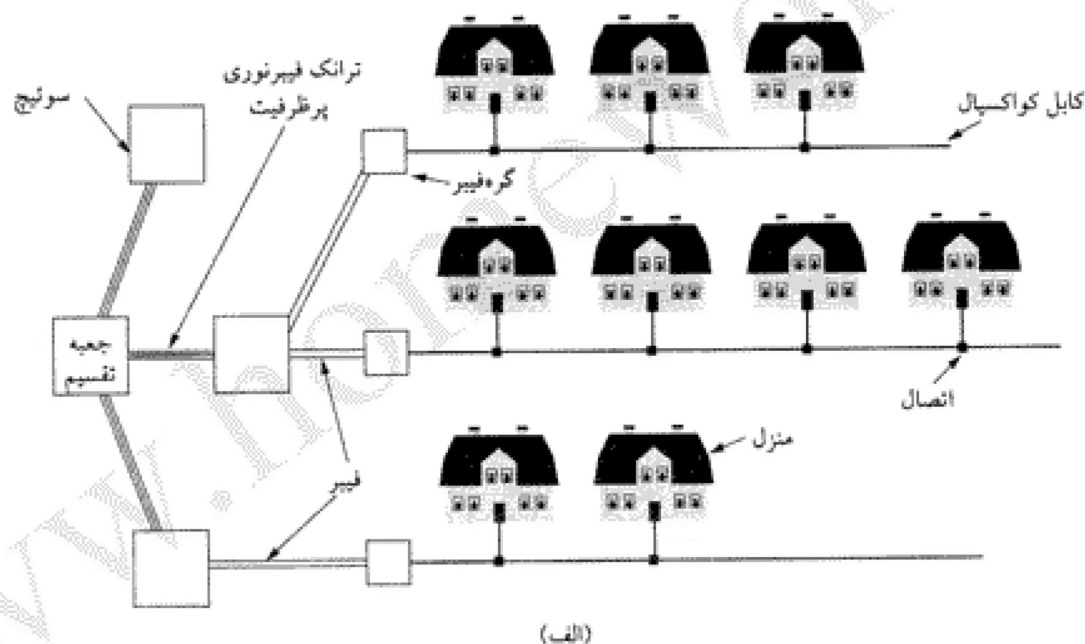


شکل ۲-۴۶. یک سیستم تلویزیون کابلی اولیه.

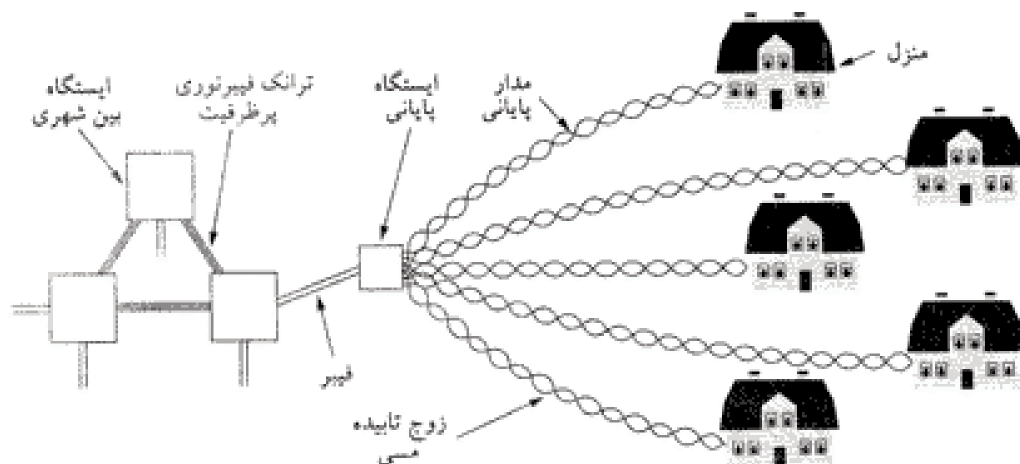
سیستم که ترکیبی بود از فیبر نوری (بین شهرها) و کابلهای کواکسیال (از مراکز توزیع تا منازل) HFC (**آمیخته فیبر-کواکس** – Hybrid Fiber Coax) نام دارد. در نقاطی که **گره فیبر** (fiber node) خوانده می‌شوند، سیگنالهای نوری به الکترونیکی (و بالعکس) تبدیل می‌شود. از آنجائیکه پهنای باند فیبر نوری بسیار بیشتر از کواکس است، یک گره فیبر می‌تواند تعداد زیادی کابل کواکسیال را تغذیه کند. در شکل ۲-۴۷(الف) یک سیستم HFC مدرن را ملاحظه می‌کنید.

در سالهای اخیر، بسیاری از شرکتهای کابلی تصمیم گرفته‌اند وارد تجارت دسترسی اینترنت (و همچنین تلفن) شوند. با این حال تفاوتهای فنی شبکه‌های کابلی و تلفن چنان است که تاکنون مانع از تحقق کامل این خواسته شده است. یکی از مشکلاتی که می‌توان بعنوان نمونه به آن اشاره کرد، تقویت‌کننده‌های یکطرفه‌ای است که باید با تقویت‌کننده‌های دوطرفه جایگزین شوند.

اما تفاوت مهمتری بین سیستم HFC شکل ۲-۴۷ (الف) و سیستم تلفن (شکل ۲-۴۷ ب) وجود دارد، که برطرف کردن آن بسیار مشکلتر است. در سیستم HFC یک رشته کابل کواکسیال بین خانه‌های متعددی مشترک



(الف)



(ب)

شکل ۲-۴۷. (الف) تلویزیون کابلی. (ب) سیستم تلفن ثابت.

است، در حالیکه در سیستم تلفن شهری هر خانه دارای یک اتصال (مدار پایانی) خاص خود است. در پخش برنامه‌های تلویزیونی این مشترک بودن کابل اهمیت چندانی ندارد: یک برنامه پخش می‌شود، و اهمیتی ندارد که ۱۰ نفر آنرا می‌بینند یا ۱۰,۰۰۰ نفر. اما وقتی پای دسترسی اینترنت به میان می‌آید، ۱۰ نفر با ۱۰,۰۰۰ نفر فرق بسیاری با هم دارند. اگر یکی از کاربران بخواهد فایل بزرگی را از اینترنت بگیرد، پهنای باند لازم برای این کار از کاربران دیگر گرفته می‌شود. هر چه تعداد کاربران بیشتر شود، رقابت بر سر پهنای باند شدیدتر خواهد بود. در سیستم تلفن چنین مشکلی وجود ندارد: اگر شما مشغول گرفتن یک فایل بزرگ روی خط ADSL خود باشید، تأثیری روی کار همسایه‌تان نخواهد گذاشت. از طرف دیگر، پهنای باند یک کابل کواکس بسیار بیشتر از زوج-تابیده است.

شرکتهای کابلی برای غلبه بر این مشکل، کابلهای بلند را تکه تکه کرده و هر کدام را مستقیماً به یک گره فیبر متصل می‌کنند. اگر تعداد کاربران هر کابل کواکس خیلی زیاد نباشد، پهنای باند فیبر نوری عملاً نامحدود خواهد بود و می‌توان ترافیک را بخوبی مدیریت کرد. امروزه هر کابل کواکس به ۵۰۰ تا ۲۰۰۰ نفر سرویس می‌دهد، ولی با زیاد شدن تعداد مشترکین این سیستم شاید لازم باشد از کابلها و گره‌های فیبر بیشتری استفاده کرد.

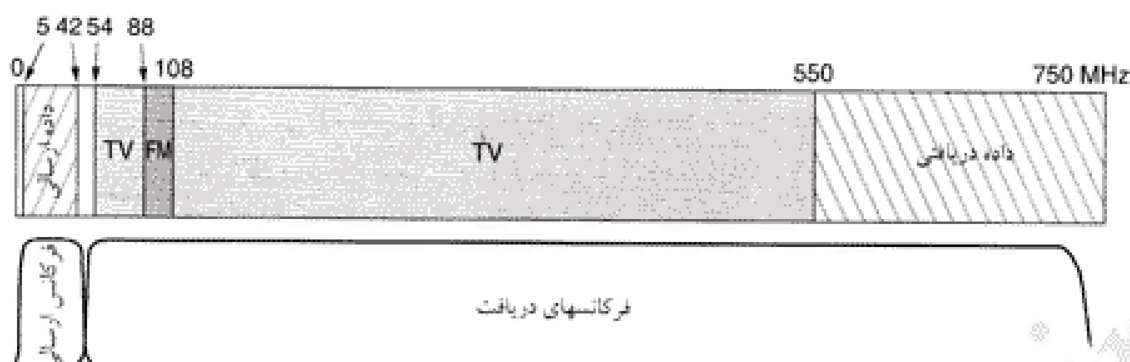
۳-۷-۲ تخصیص طیف فرکانسی

حذف تمام کانالهای تلویزیونی و بکارگیری زیرساخت‌های کابلی برای دسترسی اینترنت باعث نارضایتی تعداد زیادی از مشترکین خواهد شد، و بهمین دلیل شرکتهای کابلی در این کار تردید دارند. علاوه بر آن، در اغلب شهرها قوانین سختگیرانه‌ای برای کنترل آنچه روی کابلها فرستاده می‌شود، وجود دارد، و حتی اگر شرکتهای کابلی واقعاً هم بخواهند اجازهٔ چنین کاری را ندارند. در نتیجه، آنها باید راهی برای همزیستی تلویزیون کابلی و اینترنت روی یک کابل می‌یافتند.

کانالهای تلویزیون کابلی در منطقهٔ آمریکای شمالی در ناحیهٔ 54-550 MHz طیف قرار دارند (البته باستثنای محدودهٔ 88-108 MHz که به رادیوی FM اختصاص دارد). هر کانال (با احتساب باندهای محافظ) 6 MHz پهنا دارد. در اروپا، حد پائین طیف معمولاً 65 MHz است و کانالها نیز 6-8 MHz پهنا دارند، چون سیستمهای تلویزیونی PAL و SECAM کیفیت بالاتری دارند. از قسمت پائین این باند استفاده نمی‌شود. کابلهای جدید می‌توانند بالاتر از 550 MHz (اغلب تا 750 MHz) نیز کار کنند. راه حل انتخاب شده این بود که کانالهای ارسال به اینترنت (upstream) در باند 5-42 MHz (در اروپا، کمی بالاتر) و کانالهای دریافت از اینترنت (downstream) در فرکانسهای بالاتر تعریف شوند. در شکل ۲-۴۸ طیف فرکانسی کابل کواکسیال را ملاحظه می‌کنید.

از آنجائیکه سیگنالهای تلویزیونی فقط در یک جهت (مرکز توزیع به منازل) منتشر می‌شوند، تقویت‌کننده‌های ارسال فقط در باند 5-42 MHz ، و تقویت‌کننده‌های دریافت فقط در باند بالاتر از 54 MHz کار می‌کنند. همانطور که می‌بینید، بین باندهای ارسال و دریافت یک عدم‌تقارن بوجود آمده، و باند دریافت از اینترنت بسیار وسیعتر است (البته این وضعیت چندان هم بد نیست، چون اغلب ترافیک اینترنت در همین جهت است). قبلاً هم دیدید که شرکتهای تلفن عمداً (و بدون اینکه اجبار تکنیکی وجود داشته باشد) سرویس DSL را بصورت نامتقارن درمی‌آورند.

توانایی کابلهای بلند کواکسیال در انتقال سیگنالهای دیجیتال چندان بهتر از سیمهای زوج-تابیده نیست، و به همین دلیل آنها هم به نوعی مدولاسیون آنالوگ احتیاج دارند. برای مدولاسیون هر کانال 6 MHz (یا 8 MHz) از روش QAM-64 (و اگر کیفیت کابل فوق‌العاده خوب باشد، از QAM-256) استفاده می‌شود. با یک کانال 6-MHz و مدولاسیون QAM-64 ، نرخ انتقال داده‌ای معادل 36 Mbps بدست می‌آید، که اگر سرآیند از آن کسر شود، تقریباً 27 Mbps باقی می‌ماند. با مدولاسیون QAM-256 نرخ انتقال داده (بعد از کسر سرآیند) 39 Mbps



شکل ۲-۴۸. تخصیص فرکانس در یک سیستم تلویزیون کابلی برای دسترسی اینترنت.

خواهد بود. (ظرفیت سیستمهای اروپایی 1/3 بیشتر است.)

در کانالهای ارسال به اینترنت بدلیل وجود نویز بالا (از منابع مایکروویو زمینی، و رادیوهای محلی) حتی QAM-64 نیز بخوبی کار نمی‌کند، و باید از روشهای محافظه کارانه‌تری (مانند QPSK) استفاده کرد. روش QPSK ظرفیت بسیار کمتری نسبت به مدولاسیون QAM دارد (شکل ۲-۲۵ را ببینید) - 2 bits/baud بجای 6-8 bits/baud . با این اوصاف، عدم تقارن کانالهای ارسال و دریافت حتی از آنچه در شکل ۲-۴۸ دیده می‌شود، نیز بیشتر است.

شرکتهای کابلی مجبور شدند تقویت‌کننده‌های سادهٔ خود را نیز با سیستمهای دیجیتالی هوشمند جایگزین کنند. نام این تجهیزات نیز از **جعبه تقسیم** (headend) به CTMS (**سیستم پایانه‌ای مودم کابلی** - Cable Modem Termination System) تغییر کرده است، ولی ما در قسمتهای آتی همچنان از اصطلاح «جعبه تقسیم» برای اشاره به این تقویت‌کننده‌ها استفاده خواهیم کرد.

۲-۷-٤ مودمهای کابلی

مودم کابلی (cable modem) دستگاهیست با دو سر: یک سر به کامپیوتر وصل می‌شود، و سر دیگر به شبکهٔ کابلی. در سالهای اولیه اینترنت کابلی، هر شرکت مودم کابلی خاصی داشت، که توسط تکنسینهای آن نصب می‌شد. اما بزودی معلوم شد که وجود سیستمی با استاندارد باز موجب گسترش رقابت در بازار، کاهش قیمتها، و در نتیجه اقبال عمومی به این سرویسها خواهد شد. علاوه بر آن، خریدن یک مودم از فروشگاه و نصب آن توسط خود مشتری بسیار ساده‌تر از مراجعهٔ یک تکنسین برای نصب هر مودم است (که هزینهٔ بسیار بالایی نیز دارد).

در نتیجه، شرکتهای کابلی بزرگ برای تولید یک مودم کابلی استاندارد (و تست سازگاری آنها) با شرکتی بنام CableLabs متحد شدند. این استاندارد که DOCSIS (Data Over Cable Service Interface Specification) نام داشت، فقط نقطهٔ شروعی بود برای جایگزینی مودمهای متنوع و ناسازگار - ویرایش اروپایی این استاندارد نیز EuroDOCSIS نام گرفت. اما ایدهٔ استاندارد کردن مودمهای کابلی برای همهٔ شرکتهای کابلی جالب نبود، چون آنها پول خوبی بابت اجاره دادن مودم به علاقمندان این سرویسها به جیب می‌زدند. یک استاندارد باز پای دهها تولیدکنندهٔ جدید را به این قلمرو باز می‌کرد، و به این تجارت پُر منفعت پایان می‌داد.

ارتباط مودم به کامپیوتر ساده و سرراست است: در حال حاضر از اترنت 10-Mbps (و گاهی هم USB) برای این منظور استفاده می‌شود. بطور مسلم در آینده شاهد کارتهای مودم کابلی (شبیه کارتهای V.9x فعلی) نیز خواهیم بود.

ارتباط در سر دیگر پیچیده‌تر است، و بخش عمده‌ای از استاندارد آن به مهندسی رادیو مربوط می‌شود (که از

حوصلهٔ این کتاب خارج است). تنها نکته‌ای که باید بیاد داشته باشید این است که، یک مودم کابلی (مانند مودمهای ADSL) همیشه روشن و متصل است. ارتباط این مودمها به محض روشن شدن برقرار شده و تا زمان خاموش شدن در همین حالت باقی می‌ماند (هزینهٔ آنها نیز بر حسب زمان محاسبه نمی‌شود).

برای درک بهتر طرز کار مودمهای کابلی، اجازه دهید ببینیم با روشن کردن آنها چه اتفاقی می‌افتد. مودم بمحض روشن شدن تمام کانالهای دریافت را برای پیدا کردن بستهٔ خاصی که در فواصل منظم از طرف منبع ارسال می‌شود (و حاوی پارامترهای سیستم برای مودمهایی که تازه روشن شده‌اند، می‌باشد) جستجو می‌کند. پس از پیدا کردن این بسته، مودم جدید حضور خود را از طریق یکی از کانالهای ارسال به منبع اعلام می‌کند. منبع نیز با تخصیص کانالهای ارسال و دریافت پاسخ مودم را می‌دهد. البته این کانالها هر زمان که منبع لازم ببیند (مثلاً، برای متعادل کردن بار)، می‌توانند عوض شوند.

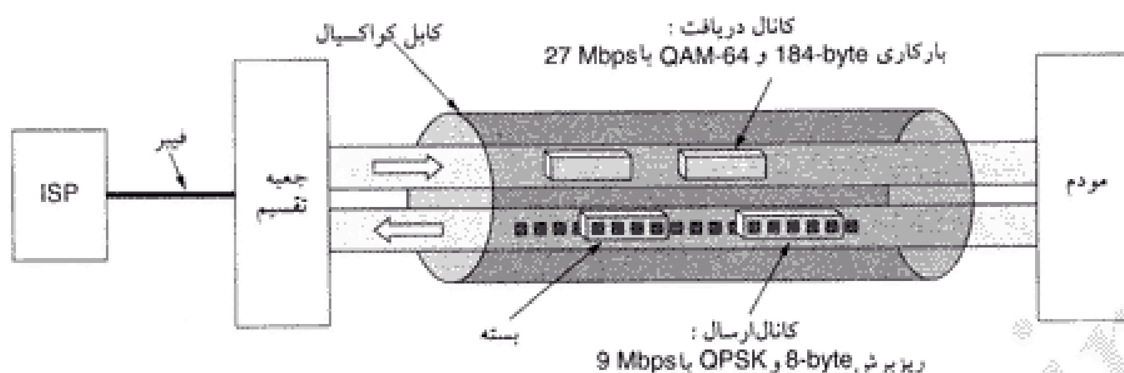
سپس مودم با ارسال یک بستهٔ خاص به منبع و محاسبهٔ زمان رفت و برگشت آن، فاصلهٔ خود با منبع را تعیین می‌کند؛ به این کار **تعیین فاصله** (ranging) گفته می‌شود. این کار برای کارکرد صحیح کانالهای ارسال و همزمانی لازم است. این فاصلهٔ زمانی به **ریزبُرشها** (minislots) تقسیم می‌شود: هر بسته باید دقیقاً در یک یا چند ریزبُرش متوالی جا شود. منبع در فواصل منظم دور جدیدی از ریزبُرشها را اعلام می‌کند، ولی (بدلیل یکسان نبودن فاصلهٔ مودمها از منبع) این شلیک شروع مسابقه همزمان در تمام مودمها شنیده نخواهد شد. هر مودم، با دانستن فاصلهٔ خود از منبع، می‌تواند بفهمد که اولین ریزبُرش واقعاً در چه زمانی شروع شده است. طول ریزبُرشها به شبکه بستگی دارد (و ظرفیت آن معمولاً معادل ۸ بایت است).

در حین آماده‌سازی اولیه، منبع به هر مودم یک ریزبُرش تخصیص می‌دهد که می‌تواند از آن برای درخواست پهنای باند ارسال استفاده کند. قاعدتاً تعداد مودمها از ریزبُرشها بیشتر است و یک ریزبُرش به چند مودم اختصاص داده می‌شود، که این می‌تواند موجب کشمکش بین آنها شود. وقتی کامپیوتر اطلاعاتی را برای ارسال به مودم می‌فرستد، مودم تعداد ریزبُرشهای لازم را محاسبه کرده و درخواست تخصیص آنها را از منبع می‌کند. اگر این درخواست پذیرفته شود، منبع ریزبُرشهای تخصیص داده شده را (روی یک از کانالهای دریافت) به مودم اعلام می‌کند. مودم نیز در ریزبُرشهای اختصاص یافته اطلاعات خود را ارسال می‌کند (و اگر به ریزبُرشهای بیشتری نیاز داشت، می‌تواند آنها را از طریق یکی از فیلدهای سرآیند درخواست کند).

اگر جوابی از منبع نرسید (که درخواست همزمان چند مودم می‌تواند یکی از علتهای آن باشد)، مودم مدتی صبر کرده و دوباره اقدام خواهد کرد. اگر اقدام دوباره هم با شکست مواجه شد، مودم زمان انتظار را بیشتر می‌کند (بیاد دارید که این همان روش تخصیص بُرش زمانی در شبکهٔ ALOHA است؛ از تکنیکهای اترنت نمی‌توان استفاده کرد، چون مودم راهی برای گوش کردن به رسانه و تشخیص تصادم ندارد. در فصل ۴ این مبحث را مفصلتر بررسی خواهیم کرد.)

مدیریت کانالهای دریافت از اینترنت بکلی متفاوت است. اول اینکه، در اینجا فقط یک فرستنده وجود دارد (منبع)، و بالطبع کشمکشی هم در میان نیست و منبع می‌تواند از تکنیکهای مالتی‌پلکس تقسیم زمانی آماری استفاده کند. دیگر اینکه، ترافیک دریافت از اینترنت معمولاً بسیار بیشتر از ترافیک ارسال به اینترنت است، و به همین دلیل از بسته‌هایی با طول ثابت (۲۰۴ بایت) استفاده می‌شود. قسمتی از این بسته سرآیندهاست (کُد تصحیح خطای Reed-Solomon و مانند آن)، و در نهایت ۱۸۴ بایت برای داده‌های کاربر باقی می‌ماند. این عدد برای سازگاری با تلویزیون دیجیتالی MPEG-2 انتخاب شده، بنابراین فرمت کانالهای دریافت از اینترنت و تلویزیون یکسان است (شکل ۲-۴۹ را ببینید).

به فرآیند آماده‌سازی مودم برگردیم: همین که مودم فاصلهٔ خود را با منبع مشخص کرد، و کانالهای ارسال و



شکل ۲-۴۹. کانالهای ارسال و دریافت در منطقهٔ آمریکای شمالی.

دریافت و ریزبُرشها را گرفت، آماده است تا اطلاعات خود را بفرستد. اولین بسته‌ای که مودم می‌فرستد، بستهٔ خاصیست حاوی درخواست یک آدرس IP از ISP با استفاده از پروتکل DHCP (که در فصل ۵ دربارهٔ آن توضیح خواهیم داد). همچنین وقت دقیق نیز از منبع پرسیده شد، و منبع به آن جواب می‌دهد.

در قدم بعد ایمنی اطلاعات بایستی تضمین شود. از آنجائیکه کابل مودم بین تعداد زیادی از افراد مشترک است، هر کسی می‌تواند اطلاعات تمام کاربران دیگر را بخواند. برای جلوگیری از سرک کشیدن دیگران، ترافیک در هر دو جهت (ارسال و دریافت) رمز می‌شود. بخشی از فرآیند آماده‌سازی مودم شامل ایجاد کلیدهای رمز است. شاید در نگاه اول این کار غیرممکن بنظر برسد: چگونه می‌توان در روز روشن و زیر نگاه هزاران غریبه کلیدهای رمز را رد و بدل کرد؟ این کار با استفاده از الگوریتمی بنام **دیفی-هلمان** (Diffie-Hellman) ممکن است، اما اجازه دهید توضیحات بیشتر را به فصل ۸ موکول کنیم.

در پایان، مودم هویت منحصربفرد (شامل نام کاربر و کلمهٔ عبور) خود را روی کانالی که اکنون از امنیت کافی برخوردار است، به ISP اعلام می‌کند. در اینجا فرآیند آماده‌سازی به پایان می‌رسد، و کاربر می‌تواند به کار عادی خود ادامه دهد.

به همین توضیح مختصر دربارهٔ مودمهای کابلی بسنده می‌کنیم، ولی می‌توانید اطلاعات تکمیلی را در (Adams and Dulchinos, 2001; Donaldson and Jones, 2001; and Dutta-Roy, 2001) بیابید.

۲-۷-۵ مودم کابلی یا ADSL ؟

کدامیک بهتر است: ADSL یا مودم کابلی؟ این مانند آنست که بپرسید کدام سیستم عامل، یا کدام زبان بهتر است. جواب به کسی که این سؤال را از او می‌پرسید، بستگی دارد. اما اجازه دهید ADSL و مودم کابلی را از چند نقطه نظر با هم مقایسه کنیم. هر دوی این سیستمها در بخش ستون فقرات از فیبر نوری استفاده می‌کنند، ولی بخش انتهایی آنها متفاوت است. مودم کابلی در بخش انتهایی (مدار پایانی) از کابل کواکسیال استفاده می‌کند، و ADSL از زوج-تابیده. از نظر تئوری، ظرفیت کواکس صدها برابر زوج-تابیده است. اما، تمام این ظرفیت در اختیار کاربر اینترنت نیست، چون پهنای باند کابل صرف چیزهای بدردنخوری (مثل برنامه‌های تلویزیونی) هم می‌شود.

در عمل، مقایسهٔ ظرفیت این سیستمها بسیار مشکل است. در ADSL، شرکت سرویس‌دهنده ظرفیت خط را به صراحت مشخص می‌کند (مثلاً، 1 Mbps دریافت از اینترنت و 256 kbps ارسال به اینترنت)، و در اغلب موارد ۸۰٪ این ظرفیت نیز محقق می‌شود. اما شرکتهای کابلی هرگز دربارهٔ ظرفیت سرویس خود صحبت نمی‌کنند، چون این ظرفیت به تعداد کاربرانی که در هر لحظه روی یک کابل هستند، بستگی دارد. گاهی کابل بهتر از ADSL است، و گاهی بدتر. اما چیزی را نمی‌توان از قبل پیش‌بینی کرد (و این از همه آزاردهنده‌تر است). در یک لحظه

کیفیت کابل خوب است، و لحظهٔ بعد (وقتی یکی از آن خوره‌های اینترنت کامپیوتر خود را روشن می‌کند) غیر قابل تحمل.

در ADSL افزایش تعداد کاربران تأثیری روی کیفیت خط کاربران موجود نخواهد گذاشت، چون هر کاربر دارای یک خط مستقل است. در حالیکه در کابل این افزایش موجب افت کیفیت می‌شود. تنها راه مقابله با این مشکل هم انشعاب کابلهای بیشتر از گره فیبر، و کم کردن تعداد کاربران هر خط است. اما این یعنی هزینهٔ بیشتر، چیزی که صاحبان شرکتهای کابلی در برابر آن مقاومت می‌کنند.

سیستمهای تلفن همراه هم وضعیتی شبیه مودم کابلی دارد: گروهی از کاربران بطور مشترک از یک بخش خاص از پهنای باند استفاده می‌کنند. از آنجائیکه ترافیک صدا نسبتاً یکنواخت است، تقسیم پهنای باند بین کاربران (که با تکنیکهای FDM و TDM انجام می‌شود) نیز یکنواخت و ثابت است. اما در ترافیک داده، تقسیم ثابت بهیچوجه کارایی ندارد، چون توزیع زمانی استفادهٔ هر کاربر از کانال اختصاصی (روشی که در مودم کابلی بکار می‌رود) یکنواخت نیست، و موجب هدر رفتن منابع خواهد شد. با این همه، مودم کابلی حتی از این نظر نیز بیشتر شبیه تلفن همراه است تا تلفن ثابت.

سهولت دستیابی یکی دیگر از تفاوتهای ADSL و مودم کابلی است. هر کاربری یک خط تلفن دارد، اما همهٔ آنها آنقدر به ایستگاه تلفن نزدیک نیستند که بتوانند ADSL بگیرند. از طرف دیگر، کابل نیز چیزی نیست که همه داشته باشند، اما اگر از قبل کابل دارید و شرکت طرف قرارداد شما دسترسی اینترنت هم عرضه می‌کند، باید گفت شانس آورده‌اید. در اینجا دیگر فاصله تا گره فیبر یا منبع اهمیتی ندارد. باید خاطر نشان کرد که، از آنجائیکه کابل اساساً رسانه‌ای تلویزیونی است (و از ابتدا برای این منظور نصب شده)، در اغلب شرکتها و دفاتر اداری وجود ندارد.

خطوط ADSL (بدلیل ماهیت نقطه-به-نقطهٔ آنها) ذاتاً از ایمنی بالاتری نسبت به کابل برخوردارند. هر کاربری که به کابل دسترسی داشته باشد، می‌تواند تمام بسته‌های منتشر شده روی آنرا بخواند. البته همانطور که قبلاً گفتیم، تمام شرکتهای معتبر سرویسهای کابلی ترافیک را در هر دو جهت رمز می‌کنند. اما همین که کسی بتواند اطلاعات رمز شدهٔ شما را هم بگیرد، ضریب ایمنی را پائین می‌آورد.

سیستم تلفن عموماً قابل‌اعتمادتر از کابل است. برای مثال، شرکتهای تلفن دارای سیستمهای برق اضطراری هستند، و سرویسهای آنها حتی در صورت بروز خاموشی کامل هم قطع نمی‌شود. اما در سیستمهای کابلی، اگر برق هر یک از تقویت‌کننده‌های میانی در یکی از زنجیره‌ها قطع شود، ترافیک دریافت آن خط بکلی متوقف خواهد شد.

و بالاخره اینکه، اغلب شرکتهایی که سرویس ADSL ارائه می‌کنند، انتخاب ISP را بر عهدهٔ خود شما می‌گذارند (در برخی نقاط حتی اجبار قانونی برای این کار وجود دارد). در مورد شرکتهای کابلی، اغلب خود آنها ارائه‌دهندهٔ سرویس اینترنت هم هستند، و دست شما برای انتخاب ISP باز نیست.

نتیجهٔ نهایی این بحث آنست که، ADSL و کابل بیشتر به هم شبیه‌اند تا متفاوت. آنها سرویسهای مشابهی ارائه می‌کنند، و با افزایش رقابت قیمت آنها هم احتمالاً روزبروز به یکدیگر نزدیکتر خواهد شد.

۲-۸ خلاصه

لایهٔ فیزیکی اساس تمام شبکه‌هاست. طبیعت دو محدودیت بنیادی به تمام کانالهای ارتباطی تحمیل کرده، و همین محدودیت‌هاست که پهنای باند آنها را مشخص می‌کند. این دو محدودیت عبارتند از: **حد نایکوئیست** (Nyquist limit) که با کانالهای بدون نویز سروکار دارد، و **حد شانون** (Shannon limit) که به کانالهای نویزدار مربوط می‌شود.

رسانهٔ انتقال می‌تواند **هدایت‌پذیر** (guided) باشد یا **هدایت‌ناپذیر** (unguided) . مهمترین رسانه‌های هدایت‌پذیر عبارتند از: زوج-تابیده، کابل کواکسیال، و فیبر نوری. رسانه‌های هدایت‌ناپذیر نیز عبارتند از: امواج رادیویی، مایکروویو، مادون‌قرمز، و لیزر. یکی از رسانه‌های انتقال رو به رشد ماهواره‌های مخابراتی (بویژه سیستمهای مدار پائین - LEO) هستند.

سیستم تلفن یکی از کلیدی‌ترین اجزای شبکه‌های گسترده (WAN) است، که مهمترین عناصر آن عبارتند از: مدارهای پایانی (local loop) ، ترانک‌ها (trunk) ، و سوئیچ‌ها (switch) . مدارهای پایانی مدارهای آنالوگ زوج-تابیده هستند، که برای انتقال داده‌های دیجیتال روی آنها باید از **مودم** (modem) استفاده کرد. ADSL با استفاده از تکنیکهای مدولاسیون و تقسیم مدار پایانی به کانالهای مجازی می‌تواند به سرعت 50 Mbps برسد. **مدارهای پایانی بیسیم** (WLL) یکی از تکنولوژیهای جدیدیست که در آینده از آن (بویژه از LMDS) بیشتر خواهید شنید.

ترانک‌های شبکهٔ تلفن دیجیتالی هستند، و از تکنیکهای مالتی‌پلکس (شامل FDM ، TDM و WDM) در آنها استفاده می‌شود. تکنیکهای سوئیچینگ نیز بر دو نوع **سوئیچینگ مداری** (circuit switching) و **سوئیچینگ بسته‌ای** (packet switching) است، که هر دو اهمیت زیادی دارند.

برای کاربردهایی که تحرک زیادی دارند، سیستم تلفن ثابت چندان مناسب نیست. تلفنهای همراه امروزه بطور گسترده‌ای برای ارتباطات صدا مورد استفاده قرار می‌گیرند، و در آیندهٔ نزدیک ترافیک داده نیز در آنها بحد قابل‌ملاحظه‌ای خواهد رسید. نسل اول تلفنهای همراه آنالوگ بود، که عمدتاً به سیستمهای AMPS متکی بود. نسل دوم تلفن همراه دیجیتال است، که در آن از سیستمهای D-AMPS ، GSM و CDMA استفاده می‌شود. نسل سوم تلفنهای همراه نیز دیجیتال خواهد بود، و در آن CDMA **پهن‌باند** استفاده خواهد شد.

یکی از سیستمهایی که می‌توان از آن برای کاربردهای شبکه نیز بهره گرفت، تلویزیون کابلی (که از تلویزیون با آنتن مرکزی به سیستمهای آمیخته فیبر-کواکس تکامل یافته) است. این سیستم پهنای باند بالقوه زیادی دارد، ولی پهنای باند واقعی آن به تعداد کاربران فعال (و اینکه مشغول چه کاری هستند) بستگی دارد.

مسائل

۱. ضرایب فوریه تابع $f(t) = t$ $(0 \le t \le 1)$ را محاسبه کنید .
۲. هر 1 msec از یک کانال بدون نویز 4-kHz نمونه‌برداری می‌شود. حداکثر نرخ دادهٔ این کانال چقدر است؟
۳. کانالهای تلویزیونی 6 MHz پهنا دارند. اگر از یک سیگنال دیجیتال چهارسطحی استفاده کنیم، چند bit/sec می‌توان در این کانال مخابره کرد؟ فرض کنید کانال بدون نویز است.
٤. اگر یک سیگنال باینری در کانالی 3-kHz که نسبت سیگنال به نویز آن 20 dB است، مخابره شود، حداکثر نرخ دادهٔ قابل دستیابی چقدر است؟
۵. برای آن که بتوان کاریر T1 را روی یک خط 50-kHz قرار داد، نسبت سیگنال به نویز چقدر باید باشد؟
۶. فرق ستارهٔ غیرفعال و تکرارکنندهٔ فعال در یک شبکهٔ فیبر نوری چیست؟
۷. پهنای باند موجود در طیفی به پهنای 0.1 μm در طول موج 1 μm چقدر است؟
۸. می‌خواهیم یکسری تصاویر کامپیوتری اسکن‌شده را روی یک رشته فیبر نوری بفرستیم. وضوح هر تصویر 480 × 640 پیکسل، و هر پیکسل ۲۴ بیت است. تصاویر با سرعت ۶۰ صفحه بر ثانیه اسکن می‌شوند. پهنای باند موردنیاز چقدر است؟ اگر از باند 1.30 μm استفاده کنیم، به چند میکرون از طول موج نیاز داریم؟
۹. آیا قضیهٔ نایکونیست برای فیبرهای نوری هم صادق است، یا فقط برای کابلهای مسی کاربرد دارد؟
۱۰. در شکل ۲-۶، باند سمت چپ از بقیهٔ باندها باریکتر است. چرا؟

۱۱. یک آنتن زمانی بهترین بهره را دارد که قطر آن معادل طول موج امواج رادیویی باشد. قطر قابل قبول آنتنها بین ۱ تا ۵ متر است. این قطر معادل کدام طیف فرکانسی است؟

۱۲. محوشدگی چندمسیره زمانی به حداکثر می‌رسد که دو موج با اختلاف فاز ۱۸۰ درجه وارد گیرنده شوند. برای به حداکثر رسیدن محوشدگی چندمسیره در یک لینک مایکروویو 1-GHz بطول 50-km این مقدار چقدر باید باشد؟

۱۳. پرتو لیزری بقطر 1 mm روی آشکارسازی بقطر 1 mm که روی پشت‌بامی در فاصلهٔ 100 m قرار دارد، نشانه گرفته شده است. حداکثر انحراف زاویه‌ای (بر حسب درجه) چقدر باید باشد، تا پرتو لیزری هدف را گم نکند؟

۱۴. ۶۶ ماهوارهٔ پروژهٔ ایریدیوم به شش کمربند بدور زمین تقسیم شده‌اند. در ارتفاعی که این ماهواره‌ها قرار دارند، دورهٔ گردش مداری ۹۰ دقیقه است. متوسط زمان پاس‌کاری یک فرستندهٔ زمینی بین دو ماهواره چقدر است؟

۱۵. ماهواره‌ای در مدار زمین‌ثابت با صفحهٔ استوای زمین زاویهٔ ϕ می‌سازد. آیا برای فردی که روی زمین در مدار ϕ درجهٔ شمالی ایستاده، این ماهواره در آسمان ثابت بنظر می‌رسد؟ اگر نه، حرکت آنرا توضیح دهید.

۱۶. قبل از سال ۱۹۸۴ (وقتی هر ایستگاه پایانی با کُد سه رقمی ناحیه و سه رقم اول شماره تلفن مشخص می‌شد) در سیستم تلفن چند کُد ایستگاه پایانی می‌توانست وجود داشته باشد؟ کُد ناحیه با عددی بین 2 تا 9 شروع می‌شد، رقم دوم می‌توانست 0 یا 1 باشد، و رقم سوم محدودیتی نداشت. دو رقم اول کُد محلی نیز بایستی بین 2 تا 9 ، و رقم سوم می‌توانست هر عددی باشد.

۱۷. فقط با اطلاعاتی که در اینجا بدست آوردید، آیا می‌توانید بگوئید حداکثر شماره تلفنهایی که (بدون تغییر در روش شماره‌گذاری یا اضافه کردن تجهیزات) می‌توان در ایالات متحده نصب کرد، چه تعداد است؟ آیا این تعداد شماره قابل دستیابی است؟ هر دستگاه فکس یا کامپیوتر را نیز یک تلفن در نظر بگیرید، و فرض کنید هر مشترک فقط یک دستگاه تلفن دارد.

۱۸. یک سیستم سادهٔ تلفن را که در آن دو ایستگاه پایانی و یک ایستگاه بین‌شهری بوسیلهٔ خطوط دو-طرفهٔ همزمان 1-MHz به هم متصل شده‌اند، در نظر بگیرید. هر دستگاه تلفن بطور متوسط در هر روز کاری ۸ ساعته ۴ تماس برقرار می‌کند، و زمان متوسط هر تماس ۶ دقیقه است. ده درصد تماسها راه دور هستند (یعنی از ایستگاه بین‌شهری رد می‌شوند). حداکثر تعداد شماره‌هایی که هر ایستگاه پایانی می‌تواند پشتیبانی کند، چقدر است؟ مدارها را 4-kHz در نظر بگیرید.

۱۹. یک شرکت تلفن منطقه‌ای ۱۰ میلیون مشترک دارد، که بوسیلهٔ زوج-تابیده به ایستگاه مرکزی متصل شده‌اند. متوسط طول مدارهای پایانی ۱۰ کیلومتر است. ارزش مس موجود در مدارهای پایانی این سیستم چقدر است؟ سطح مقطع سیمها را دایره‌ای بقطر 1 mm ، چگالی مس را $9.0\ gr/cm^3$ و ارزش هر کیلوگرم مس را 3 دلار در نظر بگیرید.

۲۰. یک خط لولهٔ نفت سیستمی یکطرفه است، یا دو-طرفهٔ ناهمزمان، یا دو-طرفهٔ همزمان، یا هیچکدام؟

۲۱. قیمت میکروپروسسورهای سریع آنقدر کاهش یافته، که می‌توان در هر دستگاه مودم یک میکروپروسسور قرار داد. این کار چه تأثیری روی مقابله با خطاهای خطوط تلفن دارد؟

۲۲. دیاگرام فلکی شکل ۲-۲۵ چهار نقطهٔ داده در مختصات (1, 1) ، (1, -1) ، (-1, 1) و (-1, -1) دارد. مودمی با این پارامترها در 1200 baud به چه سرعتی (bps) می‌تواند دست یابد؟

۲۳. مودمی با دیاگرام فلکی شبیه شکل ۲-۲۵ دارای نقاط داده‌ای در مختصات (0, 1) و (0, 2) است. این مودم

از مدولاسیون فاز استفاده می‌کند، یا مدولاسیون دامنه؟

۲٤. در یک دیاگرام فلکی تمام نقاط روی دایره‌ای به مرکز مبدأ مختصات واقع شده‌اند. مدولاسیون این مودم چیست؟

۲۵. یک مودم دو-طرفهٔ همزمان QAM-64 از چند فرکانس استفاده می‌کند؟

۲۶. در یک سیستم ADSL که از DMT استفاده می‌کند، 3/4 کانالهای موجود به لینک دریافت اختصاص داده شده است. هر کانال از مدولاسیون QAM-64 استفاده می‌کند. ظرفیت لینک دریافت چقدر است؟

۲۷. در سیستم LMDS چهار قطاعی شکل ۲-۳۰، هر قطاع یک کانال اختصاصی 36-Mbps دارد. طبق تئوری صف، اگر 50% کانال پُر باشد، زمان انتظار در صف معادل زمان بار شدن است. در چنین شرایطی، بار شدن یک صفحهٔ وب 5-KB چقدر طول خواهد کشید؟ بار شدن این صفحه روی یک خط ADSL با سرعت 1 Mbps چقدر طول می‌کشد؟ با یک مودم 56-kbps چقدر؟

۲۸. ده سیگنال، که هر کدام به پهنای باند 4000 Hz نیاز دارند، با استفاده از FDM روی یک کانال مالتی‌پلکس شده‌اند. حداقل پهنای باند موردنیاز این کانال چقدر است؟ پهنای باندهای محافظ را 400 Hz در نظر بگیرید.

۲۹. چرا زمان نمونه‌برداری PCM در 125 μsec ثابت شده است؟

۳۰. درصد سرآیند یک کاریر T1 (درصدی از 1.544 Mbps که بکار داده‌های کاربر نمی‌آید) چقدر است؟

۳۱. حداکثر نرخ دادهٔ یک کانال بدون‌نویز 4-kHz را با استفاده از تکنیکهای زیر مقایسه کنید:

(الف) کُدگذاری آنالوگ (مثلاً، QPSK) با 2 bits/sample .

(ب) سیستم T1 PCM.

۳۲. اگر یک سیستم T1 دچار لغزش شود، برای سنکرون شدن مجدد از اولین بیت هر فریم استفاده می‌کند. برای سنکرون شدن مجدد با احتمال خطای 0.001 ، چند فریم باید بررسی شود؟

۳۳. فرق بخش دمودلاتور یک مودم با بخش دکودر یک کُدک چیست (و آیا اساساً فرقی دارند)؟ توجه داشته باشید که هر دوی اینها سیگنالهای آنالوگ را به دیجیتال تبدیل می‌کنند.

۳٤. سیگنالی بصورت دیجیتال روی یک کانال بدون‌نویز 4-kHz (با یک نمونه در هر 125 μsec) فرستاده می‌شود. با هر یک از روشهای کُدگذاری زیر چند بیت در ثانیه ارسال می‌شود:

(الف) استاندارد CCITT 2.048 Mbps .

(ب) سیستم DPCM با مقدار نسبی سیگنال 4-bit .

(ج) مدولاسیون دلتا.

۳۵. یک سیگنال سینوسی کامل با دامنهٔ A با استفاده از مدولاسیون دلتا (با x samples/sec) کُد شده است. خروجی 1+ معادل $A/8$+ تغییر در سیگنال ورودی، و خروجی 1- معادل $A/8$- تغییر در سیگنال ورودی است. بیشترین فرکانسی که این سیستم می‌تواند بدون خطای تجمعی تعقیب کند، چقدر است؟

۳۶. ساعتهای SONET خطایی معادل 1 در 10^9 دارند. چه مدت طول می‌کشد، تا این اختلاف باندازهٔ 1 بیت شود؟ عوارض جانبی این پدیده چیست؟

۳۷. در شکل ۲-۳۷، نرخ دادهٔ کاربر OC-3 از 148.608 Mbps شروع شده است. نشان دهید این عدد چگونه از پارامترهای SONET OC-3 بدست آمده است.

۳۸. سیستم SONET برای انطباق با سرعتهای پائین‌تر از STS-1 از روشی بنام **انشعابات مجازی** (Virtual Tributaries - VT) استفاده می‌کند. یک VT عبارتست از یک بار جزئی، که می‌توان آنرا بهمراه بارهای

جزئی دیگر در یک فریم STS-1 قرار داد. VT1.5 از ۳ ستون، VT2 از ۴ ستون، VT3 از ۶ ستون و VT6 از ۱۲ ستون فریم STS-1 استفاده می‌کنند. کدام VT با هر یک از سرویسهای زیر منطبق است؟

(الف) سرویس DS-1 (1.544 Mbps).

(ب) سرویس European CEPT-1 (2.048 Mbps).

(ج) سرویس DS-2 (6.312 Mbps).

۳۹. تفاوت بنیادی سوئیچینگ مداری با سوئیچینگ بسته‌ای چیست؟

۴۰. پهنای باند کاربر یک اتصال OC-12c چقدر است؟

۴۱. سه شبکهٔ سوئیچینگ بسته‌ای هر کدام ۳ گره دارند. شبکهٔ اول دارای توپولوژی ستاره (با سوئیچ مرکزی) است، شبکهٔ دوم حلقه (دوطرفه) است، و شبکهٔ سوم اتصالات کامل داخلی دارد (یعنی هر گره مستقیماً به تمام گره‌های دیگر متصل است). بهترین، بدترین و متوسط پرش (hop) در ارتباط از یک نقطه به نقطهٔ دیگر در هر یک از این شبکه‌ها چیست؟

۴۲. زمان تأخیر ارسال یک پیام x-bit در مسیری با k پرش در یک شبکهٔ سوئیچینگ مداری و یک شبکهٔ سوئیچینگ بسته‌ای (با بار کم) را با یکدیگر مقایسه کنید. زمان برقرای مدار را s ثانیه، زمان تأخیر در هر پرش را d ثانیه، اندازهٔ هر بسته را p بیت، و نرخ انتقال داده را b pbs در نظر بگیرید. در چه شرایطی تأخیر ارسال شبکهٔ سوئیچینگ بسته‌ای کمتر است؟

۴۳. فرض کنید می‌خواهیم x بیت اطلاعات را در یک شبکهٔ سوئیچینگ بسته‌ای با k پرش، بصورت بسته‌هایی با p بیت داده و h بیت سرآیند (با این فرض که $x >> p + h$) منتقل کنیم. نرخ انتقال دادهٔ خطوط b bps، و زمان تأخیر انتشار در آنها قابل صرفنظر کردن است. چه مقداری از p تأخیر کلی را به حداقل می‌رساند؟

۴۴. در یک سیستم تلفن همراه با سلولهای شش ضلعی، استفاده از باندهای فرکانسی مشابه در سلولهای مجاور ممنوع است. اگر ۸۴۰ باند فرکانسی داشته باشیم، در هر سلول از چند فرکانس می‌توان استفاده کرد؟

۴۵. طرح کلی سلولهای یک شبکهٔ تلفن همراه بندرت مانند شکل ۲-۴۱ منظم است؛ حتی شکل هر سلول نیز منظم نیست. یک علت برای این وضعیت بیاورید.

۴۶. برای پوشش دادن به شهری با مساحت $120\ km^2$، به چه تعداد سلول PCS با قطر 100 m نیاز داریم؟ (تخمین بزنید.)

۴۷. گاهی هنگام عبور یک کاربر تلفن همراه از سلولی به سلول دیگر، با وجود اینکه تمام فرستنده‌ها و گیرنده‌ها بخوبی کار می‌کنند، ارتباط ناگهان قطع می‌شود. چرا؟

۴۸. کیفیت صدای D-AMPS بسیار پائینتر از GSM است. آیا این بخاطر اجبار D-AMPS در حفظ سازگاری با AMPS است (در حالیکه GSM چنین محدودیتی ندارد)؟ یا علت دیگری دارد؟

۴۹. حداکثر تعداد کاربران همزمان در یک سلول D-AMPS را محاسبه کنید. آیا چنین محاسبه‌ای برای GSM هم امکان دارد؟ علت را توضیح دهید.

۵۰. فرض کنید سه ایستگاه A، B و C در یک سیستم CDMA (با توالیهای چیپ شکل ۲-۴۵ب) همزمان اقدام به ارسال بیتهای 0 می‌کنند. توالی چیپ حاصله چیست؟

۵۱. در بحث متعامد بودن بردارهای توالی چیپ CDMA، گفتیم که اگر $\mathbf{S} \bullet \mathbf{T} = 0$، آنگاه $\mathbf{S} \bullet \overline{\mathbf{T}} = 0$. ثابت کنید.

۵۲. اجازه دهید متعامد بودن بردارهای توالی چیپ CDMA را به روشی دیگر بیان کنیم: هر بیت در یک جفت توالی یا یکسان هستند، یا نیستند. متعامد بودن بردارها را با استفاده از اصطلاحات یکسان بودن و یکسان نبودن توضیح دهید.

۵۳. یک گیرندهٔ CDMA توالی (1+ 1+ 3- 1- 1+ 3- 1+ 1-) را دریافت می‌کند. با فرض توالیهای چیپ شکل ۲-۴۵ (ب)، تعیین کنید کدام ایستگاه‌ها، چه بیت‌هایی را ارسال کرده‌اند؟

۵٤. شبکهٔ تلفن در بخش انتهایی دارای توپولوژی ستاره است، که در آن تمام انشعابات به ایستگاه پایانی ختم می‌شوند. در حالیکه در تلویزیون کابلی، یک کابل مشترک مانند ماری بین مشترکین مختلف خزیده است. فرض کنید در آینده در شبکه‌های کابلی بجای کابلهای مسی از فیبر نوری 10 Gbps استفاده شود. آیا با چنین سیستمی می‌توان مدل شبکهٔ تلفن (یک خط مستقل از هر مشترک به ایستگاه مرکزی) را شبیه‌سازی کرد؟ اگر پاسخ مثبت است، هر فیبر چند کاربر تلفن می‌تواند داشته باشد؟

۵۵. سیستمهای تلویزیون کابلی معمولاً دارای ۱۰۰ کانال تجاری هستند، که بطور متناوب برنامه و آگهی پخش می‌کنند. این سیستم بیشتر شبیه TDM است یا FDM ؟

۵۶. یک شرکت کابلی تصمیم می‌گیرد به ۵۰۰۰ مشترک خود سرویس اینترنت ارائه دهد. این شرکت از کابلهای کواکسیال استفاده می‌کند، که هر کابل می‌تواند تا 100 Mbps روی کانال دریافت از اینترنت ظرفیت داشته باشد. شرکت مزبور برای جذب مشتریان تصمیم می‌گیرد که تا دریافت 2 Mbps را برای هر مشترک تضمین کند. توضیح دهید این شرکت برای رسیدن به هدف فوق چه کاری باید انجام دهد؟

۵۷. با توجه به تخصیص فرکانس نشان داده شده در شکل ۲-۴۸ و اطلاعات داده شده در متن کتاب، یک سیستم کابلی چه مقدار (Mbps) از ظرفیت را به ارسال به اینترنت و چه مقدار را به دریافت از اینترنت اختصاص می‌دهد؟

۵۸. اگر سیستم کابلی کاملاً بیکار باشد، کاربر با چه سرعتی می‌تواند اطلاعات را دریافت کند؟

۵۹. مالتی‌پلکس کردن استریمهای متعدد STS-1 (که به آنها انشعاب گفته می‌شود) نقش مهمی در سیستم SONET بازی می‌کند. یک مالتی‌پلکسر 3:1 سه ورودی STS-1 را در یک خروجی STS-3 مالتی‌پلکس می‌کند. اینکار بصورت بایت به بایت انجام می‌شود، یعنی سه بایت اول خروجی بترتیب بایتهای اول انشعابهای 1 ، 2 و 3 هستند؛ سه بایت دوم خروجی بترتیب بایتهای دوم انشعابهای 1 ، 2 و 3 ؛ و الی آخر. برنامه‌ای بنویسید که این مالتی‌پلکسر 3:1 را شبیه‌سازی کند. برنامهٔ شما باید پنج روال داشته باشد: روال اصلی (که چهار روال دیگر را اجرا می‌کند)، یک روال برای هر یک از انشعابهای STS-1 (مجموعاً سه روال)، و یکی برای مالتی‌پلکسر. هر روال انشعاب یک فریم STS-1 را از فایلی بطول ۸۱۰ بایت خوانده، و این فریمها را (بایت به بایت) به روال مالتی‌پلکسر می‌فرستد. روال مالتی‌پلکسر این بایتها را خوانده، و یک فریم STS-3 را (بایت به بایت) روی خروجی استاندارد (stdout) می‌نویسد. برای ارتباط بین پردازشها از پایپ (pipe) استفاده کنید.



لایه پیوند داده

در این فصل اصول طراحی لایهٔ دوم، **لایهٔ پیوند داده** (data link layer) ، را بررسی خواهیم کرد، و طی آن با الگوریتمهای لازم برای دستیابی به یک ارتباط قابل اطمینان و کارا بین دو کامپیوتر همسایه (در لایهٔ پیوند داده) آشنا خواهید شد. منظور از دو کامپیوتر همسایه، کامپیوترهایی هستند که یک کانال ارتباطی **سیم-مانند** (کابل کواکسیال، خط تلفن، و یا ارتباط بیسیم) بین آنها برقرار است. خصلت بنیادی یک کانال «سیم-مانند» اینست که بیت‌ها دقیقاً با همان نظمی که فرستاده می‌شوند، در گیرنده دریافت شوند.

شاید در نگاه اول این خصلت آنقدر ساده و ابتدایی بنظر برسد، که فکر کنید چه نیازی به الگوریتم و نرم‌افزار هست: ماشین A بیت‌ها را می‌فرستد، و ماشین B آنها را می‌گیرد. متأسفانه، مسئله بهمین سادگی نیست، چون مدارهای مخابراتی پر از نویز و خطا هستند. علاوه بر آن ظرفیت کانالهای مخابراتی نامحدود نیست، و بین ارسال و دریافت بیت‌ها یک تأخیر زمانی نیز وجود دارد. این محدودیت‌ها تأثیر جدی روی کارایی سیستمهای انتقال داده می‌گذارند. پروتکلهای مخابراتی (که موضوع اصلی این فصل هستند) باید تمام این ملاحظات را در نظر بگیرند.

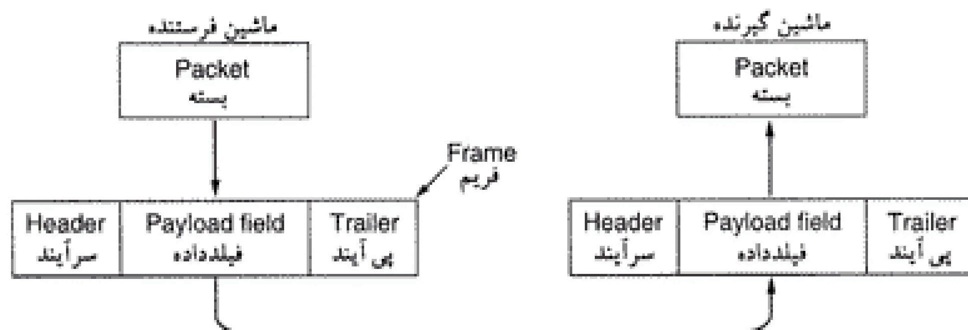
بعد از آشنایی با نکات کلیدی در طراحی لایهٔ پیوند داده، پروتکلهای این لایه را بررسی خواهیم کرد. برای شروع خصلت خطاهای کانالهای مخابراتی، منشأ آنها و نحوهٔ کشف و رفع این خطاها را بررسی کرده، و سپس پروتکلهای لازم برای حل آنها را مورد مطالعه قرار می‌دهیم. در پایان، صحت این مدلها را بررسی کرده، و چند نمونه از پروتکلهای واقعی لایهٔ پیوند داده ارائه خواهیم کرد.

۱-۳ ملاحظات طراحی لایهٔ پیوند داده

لایهٔ پیوند داده وظایف خاصی دارد که باید انجام دهد. این وظایف عبارتند از:

۱. ارائهٔ سرویسهای مشخص به لایهٔ شبکه.
۲. مدیریت خطاهای انتقال.
۳. تنظیم جریان داده‌ها (بگونه‌ایکه گیرنده‌های کُند زیر بمباران فرستنده‌های سریع غرق نشوند).

برای رسیدن به این اهداف، لایهٔ پیوند داده بسته‌های رسیده از لایهٔ شبکه را گرفته و آنها بصورت **فریم** (frame) در می‌آورد. هر فریم سه قسمت دارد: **سرآیند** (header) ، دادهٔ اصلی، و **پی‌آیند** (trailer) ؛ شکل ۳-۱ را ببینید. مدیریت فریمها کلیدی‌ترین وظیفهٔ لایهٔ پیوند داده است، که در بخشهای آینده بتفصیل دربارهٔ آن صحبت خواهیم کرد

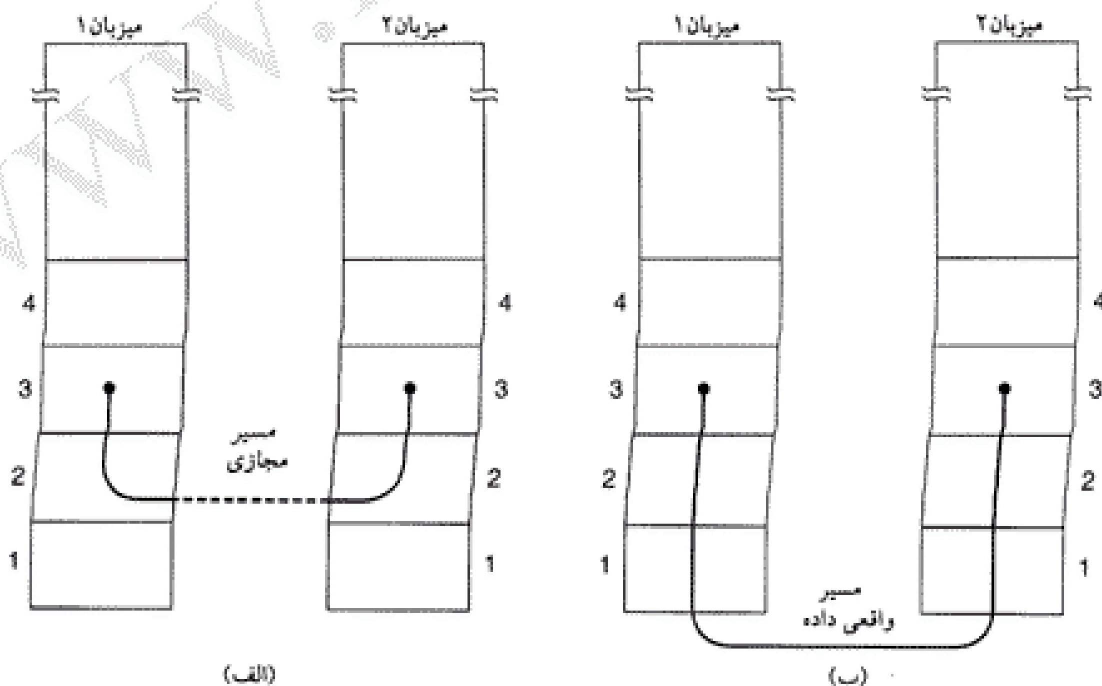


شکل ۳-۱. رابطهٔ بسته و فریم.

با اینکه این فصل منحصراً دربارهٔ لایهٔ پیوند داده و پروتکلهای آن است، اصولی که در اینجا خواهید دید (مانند کنترل خطا و کنترل جریان داده) در لایه‌های دیگر (مانند لایهٔ انتقال) نیز کاربرد دارند. در حقیقت، در بسیاری از شبکه‌ها این کارکردها را فقط در لایه‌های بالاتر (نه در لایهٔ پیوند داده) می‌توانید پیدا کنید. اما صرفنظر از اینکه آنها را کجا می‌توان پیدا کرد، اصول کار یکسان است و اهمیتی ندارد که در این فصل آنها را بررسی کنیم یا فصلهای دیگر. تنها مزیت لایهٔ پیوند داده آنست که این تکنیکها در این لایه ساده‌تر و واضحترند، بنابراین بخوبی می‌توان آنها را مطالعه کرد.

۳-۱-۱ سرویسهایی که به لایهٔ شبکه داده می‌شود

وظیفهٔ لایهٔ پیوند داده ارائهٔ سرویس به لایهٔ شبکه است. مهمترین این وظایف عبارتست از انتقال داده‌ها از لایهٔ شبکهٔ ماشین مبدأ به لایهٔ شبکهٔ ماشین مقصد. در لایهٔ شبکهٔ ماشین مبدأ چیزی هست بنام پروسس، که تعدادی بیت را به لایهٔ پیوند داده می‌دهد تا به مقصدی خاص منتقل کند. وظیفهٔ لایهٔ پیوند داده ماشین مبدأ انتقال این بیت‌ها به ماشین مقصد، و رساندن آنها بدست لایهٔ شبکهٔ مقابل است؛ شکل ۳-۲ (الف) را ببینید. البته مسیری که این بیت‌ها



شکل ۳-۲. (الف) ارتباط مجازی. (ب) ارتباط واقعی.

واقعاً طی می‌کنند، مانند شکل ۳-۲ (ب) است، ولی ساده‌ترست تصور کنیم دو پروسس در لایهٔ پیوند داده آنها را بین خود رد و بدل می‌کنند. بهمین دلیل در این فصل همه جا از مدل شکل ۳-۲ (الف) استفاده خواهیم کرد.

لایهٔ پیوند داده را می‌توان بگونه‌ای طراحی کرد که سرویسهای مختلفی ارائه کند، که این سرویسها از سیستمی به سیستم دیگر متفاوت است. معقولترین این سرویسها عبارتند از:

۱. سرویس غیرمتصل بدون تصدیق دریافت (unacknowledged connectionless).
۲. سرویس غیرمتصل با تصدیق دریافت (acknowledged connectionless).
۳. سرویس اتصال-گرا با تصدیق دریافت (acknowledged connection-oriented).

اجازه دهید این سرویسها را یکی یکی بررسی کنیم.

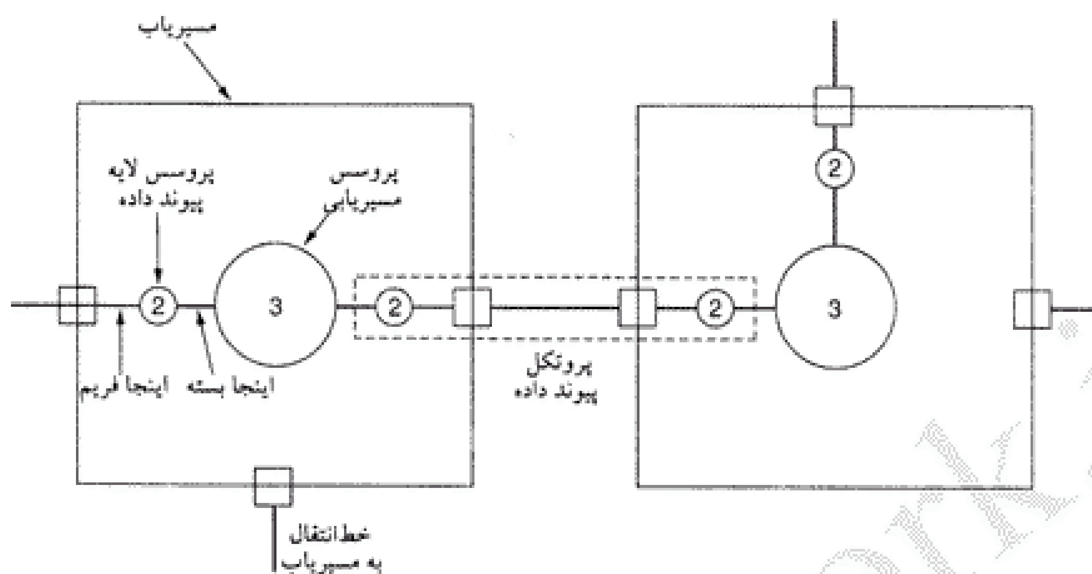
در سرویس **غیرمتصل بدون تصدیق دریافت** ماشین مبدأ فریمهای مستقلی را به ماشین مقصد می‌فرستد، بدون اینکه منتظر تصدیق دریافت آنها از طرف ماشین مقصد بماند. هیچ اتصال منطقی بین دو ماشین برقرار نمی‌شود، پس نیازی به قطع اتصال هم نیست. اگر فریمی در اثر نویز خط از بین برود، هیچ کوششی برای تشخیص این موضوع و مقابله با آن در لایهٔ پیوند داده صورت نمی‌گیرد. این سرویس برای مواقعی مناسب است که نرخ خطا بسیار پائین باشد، و در این حالت مقابله با خطا به لایه‌های بالاتر واگذار می‌شود. این سرویس برای ترافیک زمان-واقعی (مانند سرویس صدا)، که در آن دیر رسیدن بدتر از نرسیدن است، نیز مناسب است. در اغلب LAN ها نیز لایهٔ پیوند داده از سرویسهای غیرمتصل بدون تصدیق دریافت استفاده می‌کند.

سرویس بعدی که قابلیت اعتماد بیشتری دارد، سرویس **غیرمتصل با تصدیق دریافت** است. در این سرویس نیز هیچ اتصال منطقی بین مبدأ و مقصد وجود ندارد، ولی دریافت فریمها از سوی ماشین مقصد تصدیق می‌شود. بدین ترتیب، فرستنده می‌تواند پی ببرد که آیا فریمها بدرستی دریافت شده‌اند یا خیر. اگر فریمی در مدت زمان معین به مقصد نرسد، می‌توان آنرا دوباره ارسال کرد. این سرویس برای کانالهای غیر قابل اعتماد (مانند سیستمهای بیسیم) مناسب است.

لازمست تأکید کنیم که توجه به تصدیق دریافت در لایهٔ پیوند داده فقط برای بهینه‌سازی سیستم است و هیچ الزامی در آن نیست، چون این کار را همیشه می‌توان در لایهٔ شبکه انجام داد. اگر تصدیق دریافت در زمان مشخص از راه نرسد، فرستنده می‌تواند بسته را دوباره ارسال کند. مشکل اینجاست که فریمها معمولاً طول مشخصی دارند، در حالیکه بسته‌ای که لایهٔ شبکه می‌فرستد چنین نیست. اگر بسته‌ای به، مثلاً، ۱۰ فریم شکسته شود، و ۲۰ درصد این فریمها در راه گم شوند، زمان ارسال بسته بسیار طولانی خواهد شد. اما اگر برای هر فریم تصدیق دریافت درخواست شود، این کار سریعتر می‌شود. در کانالهای قابل اعتماد مانند فیبر نوری، بار اضافی چنین پروتکلهای سختگیرانه‌ای در لایهٔ پیوند داده غیر ضروری است، اما در محیطهای ذاتاً پُرنویز مانند بیسیم ارزشش را دارد.

بهترین سرویسی که لایهٔ پیوند داده می‌تواند به لایهٔ شبکه بدهد، سرویس **اتصال-گرا** (connection-oriented) است. در این سرویس قبل از شروع ارسال داده از مبدأ به مقصد، یک اتصال بین آنها برقرار می‌شود. هر فریمی که روی این اتصال فرستاده می‌شود شماره‌گذاری شده است، و لایهٔ پیوند داده دریافت آنها را نیز تضمین می‌کند. همچنین تضمین می‌شود که هر فریم فقط یک بار (و به همان ترتیب ارسال) دریافت شود. اما در سرویسهای غیرمتصل، می‌توان انتظار داشت که بسته‌ای چندین بار ارسال (و در نتیجه چندین بار هم دریافت) شود. سرویس اتصال-گرا استریم قابل اعتمادی از بیت‌ها را در اختیار لایهٔ شبکه می‌گذارد.

ارسال داده‌ها در سرویس اتصال-گرا سه مرحله دارد. در مرحلهٔ اول اتصال برقرار شده، و متغیرهای لازم (برای شمارش فریمها، و اینکه کدام فریمها دریافت شده‌اند و کدامها خیر) ست می‌شوند. در مرحلهٔ دوم، فریمها منتقل می‌شوند. و در مرحلهٔ آخر، اتصال قطع شده و منابع آن (متغیرها و بافرها) آزاد می‌شود.



شکل ۳-۳. محل فعالیت لایهٔ پیوند داده.

اجازه دهید یک مثال بزنیم: یک زیرشبکهٔ WAN متشکل از چند مسیریاب که با خطوط نقطه-به-نقطهٔ تلفن به یکدیگر متصل شده‌اند، را در نظر بگیرید. وقتی یک فریم به مسیریاب می‌رسد، سخت‌افزار (با استفاده از تکنیکهایی که در همین فصل خواهید دید) آنرا از نظر خطا چک می‌کند، و سپس به نرم‌افزار لایهٔ پیوند داده (که می‌تواند روی چیپهای کارت شبکه قرار داشته باشد) تحویل می‌دهد. نرم‌افزار لایهٔ پیوند داده فریم را چک می‌کند تا مطمئن شود همان چیزیست که باید باشد، و اگر چنین بود، قسمت دادهٔ اصلی آنرا به نرم‌افزار مسیریابی می‌دهد. نرم‌افزار مسیریابی مسیر خروجی مناسب را تعیین کرده، و بسته را به لایهٔ پیوند داده پس می‌دهد تا ارسال شود (شکل ۳-۳ را ببینید.)

نرم‌افزارهای مسیریابی معمولاً حوصلهٔ بسته‌هایی که مدام گم می‌شوند را ندارند، و دوست دارند بسته‌ها درست و مرتب روی خطوط نقطه-به-نقطه تحویل شوند. این دیگر بر عهدهٔ پروتکل لایهٔ پیوند داده است که خطوط پُرنویز و غیر قابل اعتماد را بصورتی مطمئن (یا نسبتاً مطمئن) در آورد. با اینکه در شکل ۳-۳ نرم‌افزار لایهٔ پیوند داده (در هر مسیریاب) در دو نقطه دیده می‌شود، اما این در واقع یک پروسس واحد است که (به کمک جدول‌ها و ساختمان داده‌های مختلف) تمام کارها را انجام می‌دهد و تمام خطوط را کنترل می‌کند.

۲-۱-۳ فریم‌بندی

لایهٔ پیوند داده به لایهٔ شبکه سرویس می‌دهد، و خود نیز از سرویسهای لایهٔ فیزیکی استفاده می‌کند. چیزی که لایهٔ فیزیکی می‌گیرد، استریمی است از بیت‌ها که باید آنرا به طرف مقابل تحویل دهد. هیچ تضمینی وجود ندارد که این استریم سالم و عاری از خطا به مقصد برسد. تعداد بیتهای رسیده می‌تواند کمتر، مساوی یا بیشتر از بیت‌های ارسال شده باشد، و یا حتی مقدار برخی از آنها تغییر کرده باشد. این بر عهدهٔ لایهٔ پیوند داده است که خطاها را کشف کرده، و در صورت لزوم آنها را برطرف کند.

یکی از روشهای متداول اینست که استریم بیت‌ها در لایهٔ پیوند داده به چند فریم شکسته شده، و برای هر فریم **جمع تطبیقی** (checksum) محاسبه شود. (الگوریتمهای جمع تطبیقی را در همین فصل خواهید دید.) وقتی فریمها به مقصد می‌رسند، جمع تطبیقی آنها مجدداً محاسبه شده و با جمع تطبیقی مبدأ (که به انتهای فریم ضمیمه شده) مقایسه می‌شود. اگر این دو یکی نباشند، لایهٔ پیوند داده متوجه می‌شود که خطایی در فریم رخ داده، و به

سراغ روشهای مقابله با خطا می‌رود (که یکی از این روشها می‌تواند دور انداختن فریم، و درخواست ارسال مجدد آن باشد).

شکستن استریم بیت‌ها به فریم (که به آن **فریم‌بندی** - framing - گفته می‌شود) از آنچه در نگاه اول بنظر می‌رسد، مشکلتر است. یکی از روشهای فریم‌بندی می‌تواند انداختن فاصلهٔ زمانی در نقاطی از استریم بیت‌ها باشد (مانند فاصله انداختن بین کلمات متن). ولی در شبکه‌ها بندرت زمانبندی وجود دارد، و امکان دارد این فاصله‌ها از بین بروند و یا بیشتر شوند.

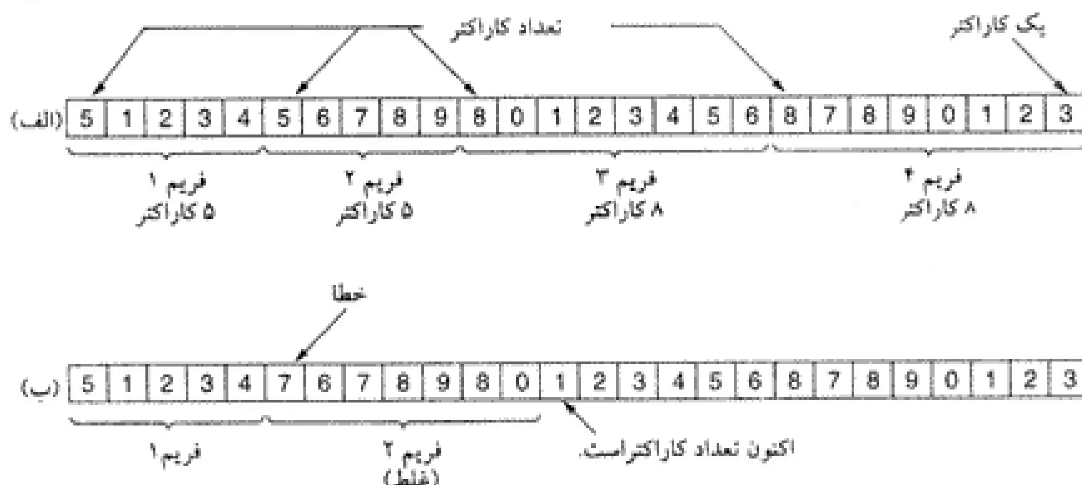
از آنجائیکه تکیه بر زمانبندی برای تعیین ابتدا و انتهای فریمها بسیار خطرناک است، روشهای دیگری برای اینکار ابداع شده، که در این قسمت با چهار تا از آنها آشنا می‌شوید:

۱. **شمارش کاراکترها.**
۲. **بایت‌های پرچم، با لاگذاری بایت.**
۳. **پرچمهای شروع و پایان، با لاگذاری بیت.**
٤. **حالتهای غیرمجاز کُدگذاری لایهٔ فیزیکی.**

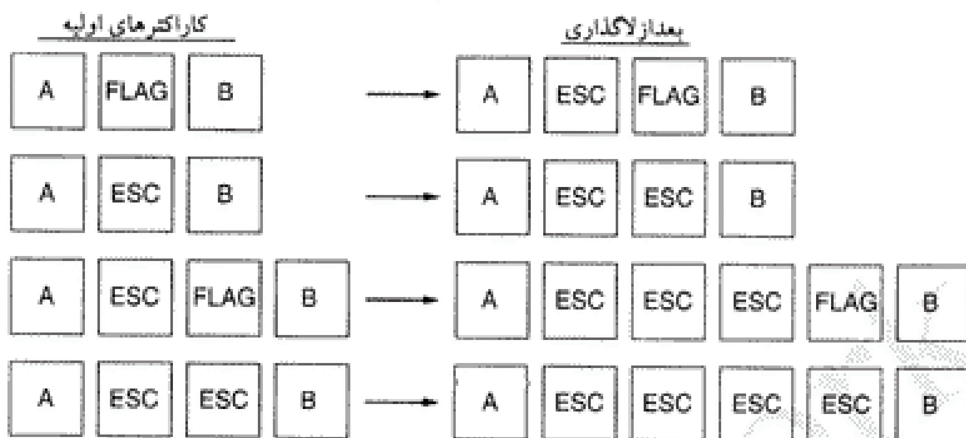
در اولین روش فریم‌بندی تعداد کاراکترهای فریم در یکی از فیلدهای سرآیند آن نوشته می‌شود. وقتی این فریم به مقصد می‌رسد، لایهٔ پیوند داده می‌تواند به کمک این فیلد ابتدا و انتهای فریم را مشخص کند. در شکل ۳-۴(الف) چهار فریم با تعداد کاراکترهای ۵، ۵، ۸ و ۸ را می‌بینید.

اشکال این روش آنست که فیلد تعداد کاراکترها نیز می‌تواند دچار خطا شود. برای مثال در شکل ۳-۴(ب)، فیلد تعداد کاراکترها در فریم دوم از ۵ به ۷ تبدیل شده است، و ماشین مقصد دیگر قادر نیست فریمهای بعدی را بدرستی بخواند (چون قادر نیست ابتدای آنها را تشخیص دهد). حتی اگر جمع تطبیقی اشتباه باشد و ماشین مقصد متوجه باشد که خطایی رخ داده، باز هم تشخیص نقطهٔ شروع بعدی برای آن غیرممکن است. درخواست ارسال مجدد نیز کمکی نمی‌کند، چون ماشین مقصد نمی‌داند فریمها تا کجا درست بوده، و اشتباه از کجا رخ داده است و نیاز به ارسال مجدد دارد.

مشکل سنکرون شدن مجدد مبدأ و مقصد بعد از بروز خطا در روش دوم فریم‌بندی (بایت‌های پرچم، با لاگذاری بایت) حل شده است، بدین ترتیب که هر فریم با تعدادی بایت خاص شروع و پایان می‌یابد. در گذشته، بایتهای شروع و پایان متفاوت بودند. ولی در سالهای اخیر از بایتهای یکسانی بعنوان بایت پرچم (flag byte) در



شکل ۳-۴. استریم کاراکترها. (الف) بدون خطا. (ب) با خطا.



شکل ۳-۵. (الف) تعیین ابتدا و انتهای فریم با استفاده از بایت پرچم. (ب) چهار توالی بایت قبل و بعد از لاگذاری بایت.

شروع و پایان فریم استفاده می‌شود. این بایتها را در شکل ۳-۵ (الف) با عنوان FLAG ملاحظه می‌کنید. در این روش اگر گیرنده همزمانی خود با فرستنده را از دست بدهد، فقط کافیست با جستجوی بایت پرچم انتهای فریم فعلی را پیدا کند. دو بایت پرچم که پشت سر هم بیایند، بمعنای پایان یک فریم و شروع فریم بعدی هستند.

یکی از مشکلات جدی این روش آنست که طرح بیت بایت پرچم می‌تواند در داده‌های اصلی نیز وجود داشته‌باشد (بویژه اگر اطلاعات از نوع برنامه‌های اجرایی یا اعداد اعشاری باشد). این وضعیت گیرنده را به اشتباه خواهد انداخت. یکی از راه‌های حل این وضعیت آنست که پروتکل لایهٔ پیوند داده در سمت فرستنده قبل از هر توالی بیت پرچم که در دادهٔ اصلی ظاهر می‌شود، یک **بایت گریز** (escape byte) خاص قرار دهد. لایهٔ پیوند داده مقصد این بایت‌ها را حذف کرده، و داده‌های اصلی را به لایهٔ شبکه تحویل می‌دهد. به این تکنیک **لاگذاری بایت** (byte stuffing) یا **لاگذاری کاراکتر** (character stuffing) گفته می‌شود. با این روش بایت پرچم بآسانی قابل تشخیص است، چون قبل از آن بایت گریز وجود ندارد.

اما حالا سؤال دیگری پیش می‌آید: اگر در وسط دادهٔ اصلی طرحی مشابه بایت گریز وجود داشت، چه اتفاقی می‌افتد؟ جواب اینست که قبل از این بایت هم یک بایت گریز قرار داده می‌شود؛ بعبارت دیگر دو بایت گریز پشت سر هم یعنی یک بایت گریز در دادهٔ اصلی. در شکل ۳-۵(ب) چند نمونه از حالتهایی که می‌تواند پیش آید، آورده شده است. در هر مورد آن چیزی که گیرنده می‌گیرد، دقیقاً مشابه آن چیزیست که فرستنده ارسال کرده است.

تکنیک لاگذاری بایت که در شکل ۳-۵ نشان داده شده، شکل سادهٔ آن چیزیست که در پروتکل PPP (یکی از مهمترین پروتکلهای ارتباط با اینترنت در کامپیوترهای شخصی) مورد استفاده قرار می‌گیرد. (بعداً در همین فصل دربارهٔ PPP صحبت خواهیم کرد.)

یکی از معایب بزرگ فریم‌بندی بایت‌های پرچم با لاگذاری بایت وابستگی شدید آن به کاراکترهای ۸-بیتی است، و همانطور که می‌دانید تمام کُدها ۸-بیتی نیستند (برای مثال، در استاندارد **یونی‌کُد** از کاراکترهای ۱۶-بیتی استفاده می‌شود). فرض ۸-بیتی بودن کاراکترها در مکانیزم فریم‌بندی یکی از مشکلات جدی آن محسوب می‌شود، بهمین دلیل روش جدیدی که در آن طول کاراکتر می‌تواند متغیر باشد، ابداع شده است.

در این روش جدید طول کاراکترها اهمیتی ندارد، و فریمها می‌توانند تعداد بیتهای دلخواه داشته باشند. طرز کار این تکنیک جدید چنین است: هر فریم با طرح بیت خاصی (01111110 - که در واقع یک بایت پرچم است) شروع می‌شود. هر گاه لایهٔ پیوند داده در سمت فرستنده پنج 1 پشت سر هم در دادهٔ اصلی دید، بطور خودکار یک 0 بعد از آن قرار می‌دهد. این روش، که به آن **لاگذاری بیت** (bit stuffing) گفته می‌شود، بسیار شبیه لاگذاری بایت است.

وقتی گیرنده پنج 1 متوالی ببیند که یک 0 پشت سر آنها آمده، بطور خودکار این 0 را حذف می‌کند. لاگذاری بیت نیز مانند لاگذاری بایت بکلی از دید لایهٔ شبکه در هر دو کامپیوتر پنهان (شفاف) است. اگر در دادهٔ کاربر طرح بیت 01111110 وجود داشته باشد، لایهٔ پیوند داده فرستنده آنرا به 011111010 تبدیل می‌کند، و در سمت گیرنده این 0 اضافی حذف شده و طرح بیت 01111110 به لایهٔ بالاتر تحویل داده می‌شود. به یک مثال در شکل ۳-۶ توجه کنید.

(الف) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(ب) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

بیت های لاگذاری

(ج) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

شکل ۳-۶. لاگذاری بیت. (الف) دادهٔ اولیه. (ب) داده‌ها بصورتی که روی خط فیزیکی ارسال می‌شود. (ج) داده‌ها بصورتی که در گیرنده دریافت می‌شود.

در روش لاگذاری بیت نیز محدودهٔ فریم با استفاده از پرچمهای شروع و پایان مشخص می‌شود، و گیرنده می‌تواند از آنها برای سنکرون شدن با فرستنده استفاده کند.

آخرین روش فریم‌بندی فقط در شبکه‌هایی قابل بکارگیری است که در کُدگذاری لایهٔ فیزیکی آنها نوعی **افزونگی** (redundancy) وجود داشته باشد. برای مثال در برخی از شبکه‌های LAN هر بیت داده با دو بیت فیزیکی نمایش داده می‌شود: بیت 1 با زوج بالا-پائین، و بیت 0 با زوج پائین-بالا. بدین‌ترتیب هر بیت داده دارای نوعی تغییر ولتاژ است، که تشخیص آنرا برای گیرنده ساده‌تر می‌کند. در چنین شبکه‌هایی زوج بالا-بالا و پائین-پائین برای داده‌ها استفاده نمی‌شود، و می‌توان از آنها برای مشخص کردن محدودهٔ فریمها سود برد.

لازم به ذکر است که در بسیاری از پروتکل‌های لینک داده برای اطمینان بیشتر از ترکیب روش شمارش کاراکترها با یکی دیگر از تکنیکهای گفته شده استفاده می‌شود. در این روش، انتهای فریم با استفاده از فیلد تعداد کاراکترها مشخص می‌شود، ولی فقط زمانی مورد قبول قرار می‌گیرد که جمع تطبیقی فریم نیز معتبر بوده و در این نقطه طرح بیت پایان فریم وجود داشته باشد. اگر چنین نباشد، گیرنده طرح بیت پایان فریم را در نقاط دیگر جستجو خواهد کرد.

۳-۱-۳ کنترل خطا

بعد از حل مسئلهٔ ابتدا و انتهای فریمها، نوبت به مسئلهٔ بعدی می‌رسد: چگونه می‌توان تمام فریمها را سالم و با ترتیب صحیح به مقصد رساند؟ فرض کنید فرستنده فقط فریمها را می‌فرستد و کاری ندارد که آنها به مقصد می‌رسند یا خیر. این وضعیت برای سرویسهای غیرمتصل بدون تصدیق دریافت خوب است، ولی برای سرویسهای قابل اعتماد (مانند سرویس اتصال-گرا با تصدیق دریافت) مسلماً خوب نیست.

یک سرویس قابل اعتماد باید بنحوی از رسیدن بسته‌ها به مقصد و آنچه در آنجا اتفاق می‌افتد، مطلع شود. معمولاً در این موارد پروتکل درخواست می‌کند که یک فریم کنترلی خاص (که محتوی تصدیق یا عدم تصدیق

دریافت صحیح فریمهاست) به فرستنده باز پس فرستاده شود. اگر فرستنده تصدیق مثبت دریافت کند، مطمئن می‌شود که فریم به سلامت به مقصد رسیده است. اما تصدیق منفی نشان می‌دهد که اوضاع روبراه نیست، و فریم باید مجدداً فرستاده شود.

مشکل دیگر اینجاست که گاهی (در اثر اشکالات سخت‌افزاری) یک فریم بکلی گم و ناپدید می‌شود. در این حالت گیرنده هیچ عکس‌العملی نشان نمی‌دهد، چون اساساً چیزی نگرفته که عکس‌العمل نشان دهد. بروشنی پیداست که در این حالت پروتکل سمت فرستنده تا ابد منتظر دریافت تصدیق از گیرنده می‌شود، تصدیقی که هرگز نخواهد رسید.

این مشکل را می‌توان با تعبیهٔ یک تایمر در لایهٔ پیوند داده حل کرد. وقتی فرستنده فریمی را می‌فرستد، تایمر را هم راه‌اندازی می‌کند. زمانی که این تایمر اندازه می‌گیرد آنقدر طولانی هست که بتوان با اطمینان گفت «فریم باید به مقصد رسیده، و تصدیق دریافت آن برگشته باشد». اگر همه چیز خوب پیش رفته باشد، معمولاً قبل از اینکه زمان تایمر به انتها برسد، تصدیق دریافت فریم به فرستنده برمی‌گردد (و تایمر ریست می‌شود).

اگر فریم یا پاسخ آن در راه گم شوند، تایمر در انتهای زمان مقرر اخطار می‌دهد؛ و ساده‌ترین راه حل همانا ارسال مجدد فریم است. اما ارسال چندبارهٔ فریمها این خطر را در بر دارد که چند تا از این فریمهای یکسان به مقصد برسند و به لایهٔ شبکه تحویل داده شوند. برای اجتناب از این وضعیت، فرستنده به هر فریم یک شمارهٔ ترتیبی می‌دهد تا گیرنده بتواند فریمهای مشابه و تکراری را تشخیص دهد.

با تمام این تمهیدات (تایمر و شمارهٔ ترتیبی فریمها) می‌توان مطمئن بود که از هر فریم یک (و فقط یک) نسخه به لایهٔ شبکه می‌رسد - و این یکی از مهمترین وظایف لایهٔ پیوند داده است. در ادامهٔ این فصل خواهید دید که لایهٔ پیوند داده چگونه این وظیفه را انجام می‌دهد.

۴-۱-۳ کنترل جریان

یکی دیگر از مسائل مهم در طراحی لایهٔ پیوند داده (و لایه‌های بالاتر) اینست که با فرستنده‌هایی که سریعتر از توان دریافت گیرنده مبادرت به ارسال اطلاعات می‌کنند، چه باید کرد؟ اگر کامپیوتر طرف فرستنده قویتر از گیرنده باشد (و یا بار کاری کمتری داشته باشد)، این وضعیت براحتی می‌تواند پیش بیاید. در این حالت گیرنده در سیلاب فریمهای ارسالی از فرستنده غرق می‌شود. حتی اگر کانال ارتباطی کاملاً عاری از خطا باشد، لحظه‌ای می‌رسد که گیرنده دیگر قادر به پردازش فریمهای ارسال شده نیست، و برخی از آنها را از دست می‌دهد. روشن است که باید کاری برای جلوگیری از این وضعیت کرد.

دو رهیافت برای مقابله با این وضعیت بکار گرفته می‌شود. در رهیافت اول، که **کنترل جریان بر اساس بازخور** (feedback-based flow control) نام دارد، این گیرنده است که آمادگی خود را برای دریافت اطلاعات بیشتر به فرستنده اعلام می‌کند (یا حداقل اعلام می‌کند در چه وضعیتی است). در رهیافت دوم، **کنترل جریان بر اساس نرخ** (rate-based flow control) ، پروتکل مکانیزمی دارد که بدون استفاده از بازخور گیرنده نرخ ارسال اطلاعات را محدود می‌کند. در این فصل با روشهای کنترل جریان بر اساس بازخور آشنا می‌شوید، ولی از آنجائیکه رهیافت دوم هرگز در لایهٔ پیوند داده کاربرد ندارد، توضیح دربارهٔ آنرا به فصل ۵ موکول می‌کنیم.

انواع مختلفی از کنترل جریان بر اساس بازخور وجود دارد، ولی همهٔ آنها اصول مشترکی دارند: پروتکل قواعد تعریف شده‌ای دارد که زمان ارسال فریم بعدی را مشخص می‌کنند. طبق این قواعد فرستنده نمی‌تواند فریم بعدی را بفرستد، مگر اینکه (بطور صریح یا ضمنی) اجازهٔ گیرنده را دریافت کرده باشد. مثلاً، وقتی اتصال برقرار می‌شود، گیرنده می‌تواند به فرستنده بگوید: «اکنون می‌توانی n فریم بفرستی، ولی بعد از آن تا اجازه نداده‌ام چیزی نفرست.»

۲-۳ کشف و تصحیح خطا

همانطور که در فصل ۲ دیدید، سیستم تلفن سه بخش عمده دارد: سوئیچها، ترانک‌ها، و حلقه‌های محلی. در اکثر کشورهای توسعه یافته دو بخش اول تماماً دیجیتال هستند. اما قسمت اعظم حلقه‌های محلی کماکان آنالوگ است، که بایستی با صرف هزینه‌های هنگفت در آینده به دیجیتال تبدیل شود. با اینکه در بخش دیجیتال خطا بندرت روی می‌دهد، نرخ آن در حلقه‌های محلی آنالوگ همچنان بالاست. علاوه بر آن، مخابرات بیسیم نیز بسرعت گسترش می‌یابد، که نرخ خطا در این قبیل سیستمها چندین برابر کانالهای فیبر نوری است. نتیجهٔ اخلاقی: فعلاً تا مدتها باید با خطاهای انتقال در سیستمهای مخابراتی بسازیم. در این قسمت خواهید دید چگونه.

خصلت خطا به منبع آن بستگی دارد؛ برای مثال، در سیستمهای رادیویی خطا بصورت **فورانی** (burst) رخ می‌دهد، نه تکی. این نوع خطا مزایا و معایبی دارد. توجه داشته باشید که کامپیوترها اطلاعات خود را بصورت بسته‌ای ارسال می‌کنند. اگر هر بستهٔ داده ۱۰۰۰ بیت و نرخ خطا نیز ۱ در ۱۰۰۰ باشد، می‌توان انتظار داشت که (در حالت غیرفورانی) تقریباً تمام بسته‌ها با خطا به مقصد برسند. اما اگر خطا بصورت فورانهای ۱۰۰ بیتی رخ دهد، بطور متوسط فقط یک یا دو بسته را خراب خواهد کرد. عیب بزرگ خطاهای فورانی آنست که کشف و تصحیح خطا در آنها بسیار دشوارتر است.

۱-۲-۳ کُدهای تصحیح خطا

طراحان شبکه دو استراتژی کلی برای مقابله با خطاها توسعه داده‌اند. یک راه اضافه کردن اطلاعات پراکنده به هر بلوک از داده‌هاست، بطوریکه گیرنده بتواند دادهٔ واقعی را از آن استنتاج کند. در روش دیگر فقط آنقدر اطلاعات اضافی به دادهٔ اصلی اضافه می‌شود که گیرنده از وقوع یا عدم وقوع خطا آگاهی یابد، و در صورت لزوم تکرار ارسال را خواستار شود. استراتژی اول **کُدهای تصحیح خطا** (error-correcting codes) و استراتژی دوم **کُدهای کشف خطا** (error-detecting codes) نام دارند. به کاربرد کُدهای تصحیح خطا اغلب **تصحیح پیشگیرانهٔ خطا** نیز گفته می‌شود.

هر یک از این تکنیکها جایگاه خاص خود را دارند. در کانالهای قابل اطمینان، مانند فیبر نوری، مقرون‌بصرفه‌تر است که از کُدهای کشف خطا استفاده کرده و بسته‌های معدودی را که خراب می‌شوند، دوباره ارسال کنیم. اما در کانالهایی مانند لینکهای بیسیم که پر از خطا هستند، بهتر است از تکنیکهای تصحیح خطا استفاده کرده و اجازه دهیم گیرنده خود دادهٔ واقعی را بدست آورد (چون باحتمال زیاد ارسال مجدد بسته‌ها هم عاری از خطا نخواهد بود).

برای مقابله با خطاها، ابتدا باید بدانیم خطا واقعاً چیست. معمولاً، یک فریم m بیت دادهٔ اصلی (یعنی، پیام) و r بیت دادهٔ پراکنده (یا اطلاعات چک‌کننده) دارد، که در مجموع n بیت می‌شود $(n = m + r)$. به این واحد nبیتی (داده‌های اصلی و پراکنده) اغلب **کلمهٔ کُد** nبیتی گفته می‌شود.

دو کلمهٔ کُد 10001001 و 10110001 را در نظر بگیرید: براحتی می‌توان مشخص کرد که این دو کلمه چند اختلاف دارند. در این مورد ۳ بیت اختلاف وجود دارد. برای تعیین تعداد اختلاف‌ها می‌توان دو کلمهٔ کُد را با هم XOR (OR انحصاری) کرد، و تعداد 1 ها را شمرد:

```
10001001
10110001
--------
00111000
```

به تعداد اختلافهای دو کلمهٔ کُد **فاصلهٔ همینگ** (Hamming distance) گفته می‌شود (Hamming, 1950). اهمیت این فاصله در آنجاست که می‌توان ثابت کرد برای تبدیل شدن اتفاقی دو کلمه با فاصلهٔ d، بایستی d خطای تک‌بیتی روی دهد.

در اکثر سیستمهای انتقال، تمامی 2^m حالت ممکنهٔ دادهٔ اصلی مجاز است، ولی بدلیل روش محاسبهٔ بیت‌های افزونگی، تمام 2^n حالت کلمهٔ کُد مجاز نیست. با توجه به الگوریتم محاسبهٔ بیت‌های افزونگی، می‌توان لیستی از تمام حالتهای مجاز کلمهٔ کُد بدست آورد، و از این لیست دو کلمه‌ای که کمترین فاصلهٔ همینگ را دارند، پیدا کرد. این فاصله فاصلهٔ همینگ الگوریتم یا کُد موردنظر است.

خصوصیات تصحیح خطا یا کشف خطای یک کُد به فاصلهٔ همینگ آن بستگی دارد. برای کشف d خطا، به کُدی با فاصلهٔ همینگ $d + 1$ نیاز داریم، چون با چنین کُدی هیچ d خطای تک‌بیتی وجود ندارد که بتواند یک کلمهٔ کُد مجاز را به کلمهٔ کُد مجاز دیگر تبدیل کند. اگر گیرنده کلمهٔ کُد غیرمجازی دریافت کرد، می‌تواند با اطمینان بگوید که خطایی رخ داده است. بهمین ترتیب، برای تصحیح d خطا، به کُدی با فاصلهٔ $2d + 1$ نیاز داریم، چون در این حالت کلمات کُد چنان از هم فاصله دارند که حتی با بروز d خطا، کلمهٔ کُد خراب شده هنوز نزدیکترین فاصله را با کلمهٔ کُد اصلی دارد، و تشخیص آن براحتی ممکن است.

بعنوان نمونه‌ای از کُدهای کشف خطا، کُدی با یک **بیت توازن** (parity bit) را در نظر بگیرید. این بیت توازن بگونه‌ای انتخاب می‌شود که تعداد بیت‌های 1 کلمهٔ کُد همواره زوج (یا فرد) شود. برای مثال، اگر بخواهیم کلمهٔ 1011010 را با **توازن زوج** (even parity) ارسال کنیم، یک بیت 0 به انتهای آن اضافه می‌کنیم (10110100)؛ اما اگر بخواهیم همین کلمه را با **توازن فرد** (odd parity) ارسال کنیم، باید یک بیت 1 به انتهای آن اضافه کنیم (10110101). کُدی با یک بیت توازن دارای فاصلهٔ همینگ 2 است، چون هر خطای تک‌بیتی کلمهٔ کُدی با توازن اشتباه تولید می‌کند. این کُد می‌توان یک خطا در هر کلمه را آشکار کند.

بعنوان یک نمونهٔ ساده از کُدهای تصحیح خطا، کُدی را در نظر بگیرید که فقط چهار کلمهٔ کُد مجاز دارد:

0000000000,   0000011111,   1111100000,   1111111111

فاصلهٔ همینگ این کُد 5 است، بنابراین می‌تواند دو خطا را تصحیح کند. اگر گیرنده کلمهٔ کُدی بصورت 0000000111 دریافت کند، می‌داند که کلمهٔ اصلی باید 0000011111 بوده باشد. اما اگر سه خطا کلمهٔ 0000000000 را به 0000000111 تبدیل کرده باشد، دیگر نمی‌توان آنرا بدرستی تصحیح کرد.

فرض کنید می‌خواهیم کُدی با m بیت دادهٔ اصلی و r بیت افزونگی طراحی کنیم که بتواند تمام خطاهای تک‌بیتی را تصحیح کند. هر یک از 2^m پیام مجاز دارای n کلمهٔ کُد غیرمجاز است که با آن 1 فاصله دارد (این را می‌توان بسادگی از معکوس کردن هر یک از بیت‌های کلمهٔ کُد nبیتی فهمید). بنابراین هر یک از 2^m پیام مجاز به طرحی اختصاصی با $n + 1$ بیت نیاز دارد. از آنجائیکه تعداد ترکیبات ممکنهٔ کلمهٔ کُد 2^n است، بایستی داشته باشیم: $(n + 1)2^m \leq 2^n$. با قرار دادن $n = m + r$ در این رابطه، داریم: $(m + r + 1) \leq 2^r$. با داشتن m، از این رابطه حداقل بیت‌های افزونگی لازم (r) برای تصحیح خطاهای تک‌بیتی بدست می‌آید.

همینگ در یکی از مقالات خود (1950) روشی برای بدست آوردن این حداقل معرفی کرد. وی بیت‌های کلمهٔ کُد را از چپ براست شماره گذاری کرد. بیت‌هایی که توانهایی از 2 هستند (1، 2، 4، 8، 16، و غیره)، بیت‌های چک‌کننده‌اند؛ سایر بیت‌ها (3، 5، 6، 7، 9، و غیره) بیت‌های پیام (m) هستند. هر بیت چک‌کننده توازن مجموعه‌ای از بیت (از جمله خودش) را زوج (یا فرد) می‌کند. هر بیت می‌تواند در بیش از یک مجموعهٔ توازن محاسبه شود. برای دیدن اینکه کدام بیت‌های چک‌کننده در محاسبهٔ توازن بیت داده‌ای در موقعیت k دخالت دارند، k را بصورت مجموع توانهای 2 می‌نویسیم. برای مثال، $11 = 1 + 2 + 8$، و $29 = 1 + 4 + 8 + 16$. هر بیت فقط با بیتهای چک‌کننده‌ای که در موقعیت‌های بدست آمده از مجموع توانهای 2 قرار دارند، چک می‌شود (مثلاً، بیت موقعیت 11 فقط با بیت‌های چک‌کنندهٔ 1، 2 و 8 چک می‌شود).

وقتی یک کلمهٔ کُد به گیرنده می‌رسد، گیرنده یک شمارنده را 0 می‌کند. سپس تمام بیت‌های چک‌کننده (k) را

کاراکتر	ASCII	بیت های چک کننده
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

ترتیب انتقال بیت ها

شکل ۳-۷. استفاده از کُد همینگ برای تصحیح خطاهای فورانی.

از نظر توازن چک می‌کند $(k = 1, 2, 4, 8, ...)$. اگر توازن k درست نباشد، گیرنده k را به شمارنده اضافه می‌کند. اگر پس از پایان این عملیات شمارنده همچنان 0 باشد، کلمهٔ کُد صحیح تلقی و قبول می‌شود. اگر شمارنده 0 نباشد، حتماً شمارهٔ بیت خطا را نشان می‌دهد. برای مثال، اگر توازن بیت‌های چک‌کنندهٔ 1 ، 2 و 8 اشتباه باشد، بیت 11 غلط است، چون این تنها بیتی است که با بیت‌های چک‌کنندهٔ 1 ، 2 و 8 چک می‌شود. در شکل ۳-۷ چند کاراکتر اَسکی ۷-بیتی را که با کُد همینگ ۱۱-بیتی کُد شده‌اند، می‌بینید. فراموش نکنید که داده‌های اصلی در موقعیتهای 3 ، 5 ، 6 ، 7 ، 9 ، 10 و 11 قرار دارند.

کُدهای همینگ فقط می‌توانند خطاهای تک‌بیتی را تصحیح کنند. با این حال روشی وجود دارد که اجازه می‌دهد تا این کُد خطاهای فورانی را نیز تصحیح کند. در این روش k کلمهٔ کُد متوالی بصورت ماتریس (یک کلمهٔ کُد در هر سطر) چیده می‌شوند. معمولاً، این کلمات تک به تک (از چپ براست) ارسال می‌شوند. برای تصحیح خطاهای فورانی، بایستی داده‌ها را بصورت ستونی (باز هم از چپ براست) ارسال کرد. وقتی k بیت اول (ستون اول) ارسال شد، نوبت به ستون دوم (و سپس ستونهای بعدی) می‌رسد (شکل ۳-۷ را ببینید). وقتی این فریم به گیرنده رسید، ماتریس از نو (ستون به ستون) ساخته می‌شود. اگر یک خطای فورانی به طول k رخ داده باشد، حداکثر یک بیت در هر کلمهٔ کُد تغییر خواهد کرد، و از آنجائیکه کُد همینگ می‌تواند یک خطا را تصحیح کند، تمام بلوک قابل تصحیح خواهد بود. در این روش برای مصون کردن km بیت داده در مقابل خطاهای فورانی با طول k (یا کمتر)، از kr بیت چک‌کننده استفاده شده است.

۳-۲-۲ کُدهای کشف خطا

کُدهای کشف خطا در لینکهای بیسیم، که در مقایسه با سیم مسی و فیبر نوری بطور وحشتناکی نویزی هستند، کاربرد گسترده‌ای دارد. بدون این کُدها شاید اساساً نتوان چیزی روی این لینکها رد و بدل کرد. اما در سیمهای مسی و فیبرهای نوری نرخ خطا بسیار کمتر است، و تشخیص خطا و ارسال مجدد بسته‌هایی که (ندرتاً) خراب می‌شوند، کاملاً کفایت می‌کند.

بعنوان مثال، کانالی را در نظر بگیرید که نرخ خطا در آن 1 در 10^6 و خطاها غیرفورانی هستند؛ اندازهٔ هر بلوک را هم 1000 بیت فرض می‌کنیم. برای داشتن ویژگی تصحیح خطا، هر بلوک ۱۰۰۰ بیتی به ۱۰ بیت چک‌کننده نیاز دارد، بعبارت دیگر برای ارسال 1 Mb داده باید 10 kb اطلاعات افزونگی (بیت‌های چک‌کننده) را نیز بهمراه آن

بفرستیم. اما برای کشف خطا فقط یک بیت توازن در هر بلوک کافیست. در این روش بار اضافی کشف خطا + ارسال مجدد یک بلوک خراب برای 1 Mb داده فقط 2001 بیت است، که در مقایسه با 10,000 بیت کُد همینگ بسیار کمتر است.

اگر در هر بلوک از یک بیت توازن برای کشف خطا استفاده کنیم و یک خطای فورانی رخ دهد، احتمال اینکه بتوانیم خطا را کشف کنیم فقط ۵۰٪ است، که بهیچوجه قابل قبول نیست. اما با تشکیل ماتریسی با n ستون و k سطر (که در بالا توضیح دادیم) اوضاع بنحو قابل توجهی بهتر خواهد شد. در این روش برای هر ستون یک بیت توازن محاسبه، و در آخرین سطر ماتریس نوشته می‌شود. هنگام ارسال نیز این ماتریس بصورت ستونی فرستاده می‌شود. گیرنده بعد از دریافت کل ماتریس، تمام بیت‌های توازن را چک می‌کند؛ و اگر هر یک از این بیت‌ها غلط باشد، ارسال مجدد ماتریس را درخواست می‌کند. این کار تا زمانی که ماتریس بطور کامل و بدون خطای توازن به دست گیرنده برسد، تکرار خواهد شد.

روش فوق می‌تواند خطاهای فورانی با طول حداکثر n بیت را آشکار کند، چون در این حالت فقط یک بیت در هر ستون تغییر خواهد کرد. اما اگر یک خطای فورانی با طول $n + 1$ رخ دهد بگونه‌ایکه فقط بیت اول و آخر را تغییر دهد (و سایر بیت‌ها تغییر نکنند)، نمی‌توان آنرا کشف کرد، زیرا بیت اول و آخر در یک سطر قرار می‌گیرند و توازن این سطر بدون تغییر خواهد ماند. (یک خطای فورانی الزاماً بمعنای معکوس شدن تمام بیت‌ها نیست: فقط می‌توان از معکوس شدن بیت اول و آخر مطمئن بود.) اگر طول خطای فورانی خیلی زیاد باشد یا تعدادی خطای فورانی کوتاه و پشت سر هم رخ دهد، احتمال اینکه یکی از ستونها تصادفاً صاحب توازن درست شود، ۵۰٪ است، بنابراین احتمال اینکه چنین بلوکی (به اشتباه) صحیح تلقی شود، 2^{-n} خواهد بود.

با اینکه روش فوق در مواردی کفایت می‌کند، اما در عمل از روش دیگری استفاده می‌شود: **کُد چندجمله‌ای** (polynominal code)، که به CRC (**چک افزونگی چرخه‌ای** - Cyclic Redundancy Check) نیز معروفست. در کُدهای چندجمله‌ای مبنا بر این است که هر رشته یک چندجمله‌ایست با ضرایب 0 و 1. با این فرض، یک فریم k-بیتی معادلست با عبارتی k جمله‌ای، با ضرایب x^{k-1} تا x^0. این چندجمله‌ای از درجهٔ $k - 1$ است. باارزشترین بیت (منتهی‌الیه سمت چپ) ضریب x^{k-1} است، بیت بعدی ضریب x^{k-2}، و الی آخر. برای مثال، رشتهٔ 110001 دارای ۶ بیت است بنابراین نشاندهندهٔ یک شش جمله‌ایست با ضرایب 1، 1، 0، 0، 0، و 1، که می‌توان آنرا چنین نوشت: $x^5 + x^4 + x^0$.

محاسبات چندجمله‌ایها در مدول 2 (و طبق قوانین جبر میدان) انجام می‌شود. در جمع و تفریق ۲ بر ۱ نادیده گرفته می‌شود، بعبارت دیگر شبیه XOR است. برای مثال،

$$\begin{array}{r} 10011011 \\ +\ \underline{11001010} \\ 01010001 \end{array} \qquad \begin{array}{r} 00110011 \\ +\ \underline{11001101} \\ 11111110 \end{array} \qquad \begin{array}{r} 11110000 \\ -\ \underline{10100110} \\ 01010110 \end{array} \qquad \begin{array}{r} 01010101 \\ -\ \underline{10101111} \\ 11111010 \end{array}$$

تقسیم درست مانند تقسیم باینری است، با این تفاوت که تفریق‌ها در مدول 2 (مانند بالا) انجام می‌شود.

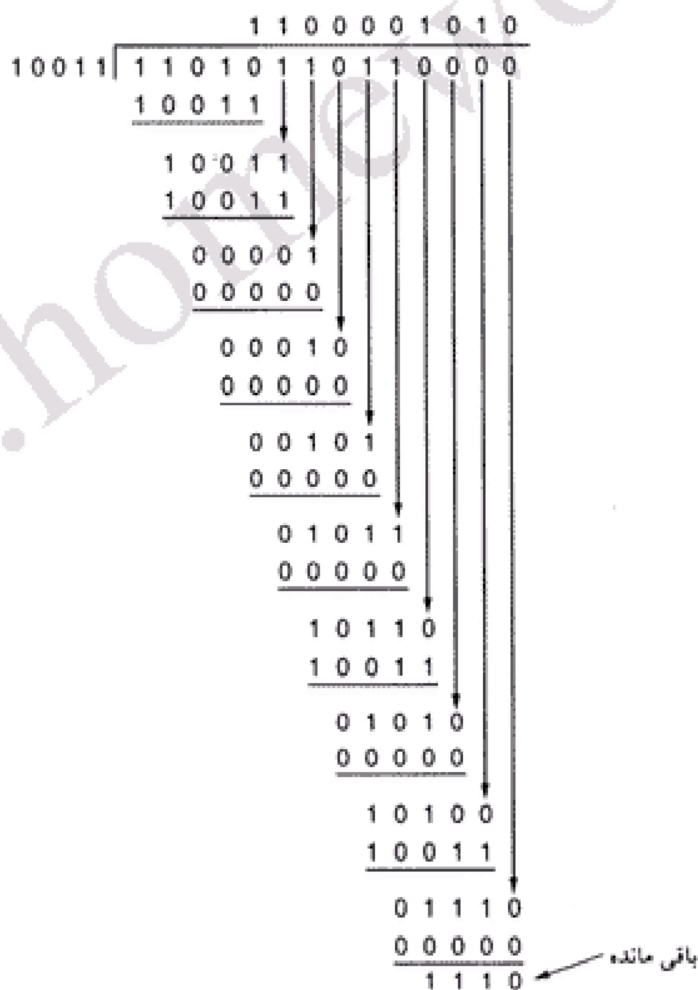
در هنگام استفاده از روش کُد چندجمله‌ای، فرستنده و گیرنده بایستی از قبل بر سر یک **چندجمله‌ای مولد** (generator polynominal)، که آنرا $G(x)$ می‌نامیم، توافق کنند. باارزشترین (چپ‌ترین) و کم‌ارزشترین (راست‌ترین) بیت‌های چندجمله‌ای مولد باید 1 باشد. برای محاسبهٔ **مجموع چک** (checksum) یک فریم m-بیتی (که چندجمله‌ای متناظر با آن $M(x)$ است)، این فریم باید طولانیتر از چندجمله‌ای مولد باشد. ایده آنست که یک مجموع چک به انتهای فریم اصلی چسبانده شود، بگونه‌ایکه فریم حاصله بر $G(x)$ قابل تقسیم باشد. اگر

تقسیم این فریم بر $G(x)$ در سمت گیرنده باقیمانده آورد، معلوم می‌شود که خطایی رخ داده است.

الگوریتم محاسبهٔ مجموع چک چنین است:

۱. فرض می‌کنیم چندجمله‌ای $G(x)$ از درجهٔ r است. r بیت 0 به سمت راست فریم اضافه می‌کنیم تا تعداد بیتهای آن به $m + r$ برسد. این چندجمله‌ای معادل $x^r M(x)$ خواهد شد.
۲. رشتهٔ $x^r M(x)$ را (در مدول 2) بر $G(x)$ تقسیم می‌کنیم.
۳. باقیمانده را (که همیشه r بیت یا کمتر دارد) از $x^r M(x)$ کم می‌کنیم (این تفریق هم در مدول 2 انجام می‌شود). حاصل تفریق همان فریم موردنظر (فریم اولیه + مجموع چک) است، که آنرا $T(x)$ می‌نامیم.

در شکل ۳-۸ طرز محاسبهٔ مجموع چک برای فریم 1101011011 را با مولد $G(x) = x^4 + x + 1$ ملاحظه می‌کنید.



شکل ۳-۸ محاسبهٔ مجموع چک کُد چندجمله‌ای.

همانطور که براحتی معلوم می‌شود، $T(x)$ بر $G(x)$ (در مدول 2) بخش‌پذیر است (چون وقتی باقیماندهٔ تقسیم را از مقسوم کم کنیم، عدد حاصله بطور حتم بر مقسوم‌علیه بخش‌پذیر خواهد بود). بطور مثال، اگر 210,278 را (در مبنای 10) بر 10,941 تقسیم کنیم، باقیمانده 2399 می‌شود که اگر آنرا از 210,278 کم کنیم، آنچه باقی می‌ماند (207,879)، بر 10,941 بخش‌پذیر خواهد بود.

حال اجازه دهید قدرت این روش را بررسی کنیم. این روش چه نوع خطاهایی را می‌تواند کشف کند؟ فرض کنید خطایی رخ داده، و بجای $T(x)$ رشتهٔ $T(x) + E(x)$ به گیرنده رسیده است، بطوریکه هر بیت 1 در $E(x)$ متناظر با یک بیت تغییر یافته است (بعبارت دیگر، اگر در این عبارت، $E(x)$، k بیت 1 وجود داشته باشد، k خطای تک‌بیتی رخ داده است). خطای فورانی نیز عبارتست از خطایی که با یک بیت 1 شروع و ختم شود، و بین آنها هر ترکیبی از 0 و 1 می‌تواند وجود داشته باشد.

وقتی این فریم به مقصد می‌رسد، گیرنده آنرا بر $G(x)$ تقسیم می‌کند (بعبارت دیگر $[T(x)+E(x)]/G(x)$ را محاسبه می‌کند). از آنجائیکه $T(x)/G(x) = 0$، این تقسیم معادل $E(x)/G(x)$ است. همانطور که می‌بینید، اگر خطای رخ داده دقیقاً طرحی شبیه $G(x)$ نداشته باشد، بطور مسلم آشکار خواهد شد.

فرض کنید یک خطای تک‌بیتی رخ داده است، یعنی، $E(x) = x^i$ (که i بیت خطاست). اگر $G(x)$ بیش از دو جمله داشته باشد، $E(x)$ هرگز بر آن بخش‌پذیر نخواهد بود – پس، این روش می‌تواند تمام خطاهای تک‌بیتی را آشکار کند.

اگر دو خطای تک‌بیتی جدا از هم رخ دهد، بطوریکه $E(x) = x^i + x^j$ (که در آن $i > j$)، می‌توان $E(x)$ را به صورت $x^j(x^{i-j} + 1)$ تجزیه کرد. اگر $G(x)$ بر x بخش‌پذیر نباشد، شرط کافی برای اینکه تمام خطاهای دوبیتی قابل کشف باشد آن است که $E(x)$ عبارت $x^k + 1$ را (برای تمام kهای کوچکتر از $i - j$) بخش نکند. چندجمله‌ای‌های ساده و از درجهٔ پائینی می‌شناسیم که می‌توان با آنها فریمهای نسبتاً طویل را محافظت کرد. مثلاً، چندجمله‌ای $x^{15} + x^{14} + 1$ هیچ عبارت $x^k + 1$ را برای تمام kهای کوچکتر از 32,768 بخش نمی‌کند.

اگر تعداد خطاهای رخ داده عددی فرد باشد، تعداد جملات $E(x)$ نیز فرد خواهد بود (برای مثال، تعداد جملات $x^5 + x^2 + 1$ فرد است، ولی $x^2 + 1$ چنین نیست). جالبست بدانید که هیچ چندجمله‌ای با تعداد جملات فرد وجود ندارد که (در مدول 2) بر $x + 1$ بخش‌پذیر باشد. بدین ترتیب اگر $G(x)$ را طوری انتخاب کنیم که بر $x + 1$ بخش‌پذیر باشد، می‌توانیم هر خطایی که تعداد بیتهای تغییر کرده فرد باشد را کشف کنیم.

برای اثبات اینکه هیچ چندجمله‌ای فرد وجود ندارد که بر $x + 1$ بخش‌پذیر باشد، فرض کنید $E(x)$ چندجمله‌ای فردیست که چنین خاصیتی دارد (بر $x + 1$ بخش‌پذیر است). اگر از $x + 1$ فاکتور بگیریم، $E(x)$ بصورت $(x + 1)\ Q(x)$ در می‌آید. حال $E(1) = (1 + 1)\ Q(1)$ را محاسبه می‌کنیم. از آنجائیکه (در مدول 2) $1 + 1 = 0$، $E(1)$ باید 0 باشد. اما اگر تعداد جملات $E(x)$ فرد باشد، قرار دادن 1 بجای x در آن همیشه نتیجهٔ 1 می‌دهد. بنابراین فرض ما نمی‌تواند درست باشد، و هیچ چندجمله‌ای فرد بر $x + 1$ بخش‌پذیر نیست.

بالاخره، و از همه مهمتر، یک کُد چندجمله‌ای با r بیت چک‌کننده تمام خطاهای فورانی با طول کمتر یا مساوی r را آشکار می‌کند. یک خطای فورانی با طول k را می‌توان با $x^i(x^{k-1} + \dots + 1)$ نشان داد، که در آن i نقطهٔ شروع خطای فورانی از سمت راست فریم است. اگر مولد $G(x)$ دارای جملهٔ x^0 باشد، بر x^i بخش‌پذیر نخواهد بود؛ بنابراین اگر درجهٔ عبارت داخل پرانتز از درجهٔ $G(x)$ کمتر باشد، باقیماندهٔ تقسیم هرگز نمی‌تواند 0 شود.

اگر طول خطای فورانی $r + 1$ باشد، باقیماندهٔ تقسیم بر $G(x)$ صفر می‌شود فقط و فقط اگر طرح بیت خطا با $G(x)$ یکسان باشد. طبق تعریف بیت‌های اول و آخر خطای فورانی باید 1 باشند، بنابراین یکسان بودن آنها به $r - 1$ بیت میانی بستگی دارد. اگر تمام ترکیبات این $r - 1$ بیت را یکسان فرض کنیم، احتمال بروز این وضعیت $\frac{1}{2^{r-1}}$ خواهد بود.

همچنین می‌توان نشان داد که اگر طول خطای فورانی از $r + 1$ بزرگتر باشد یا چند خطای فورانی کوتاهتر رخ دهد، احتمال کشف نشدن خطا (با فرض یکسان بودن تمام ترکیبات ممکنه) $\frac{1}{2^r}$ است.

برخی از چندجمله‌ایها بصورت استاندارد بین‌المللی در آمده‌اند، که از میان آنها می‌توان به چندجمله‌ای زیر (که در IEEE 802 از آن استفاده می‌شود) اشاره کرد:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$$

از ویژگیهای جالب این چندجمله‌ای آن است که هر نوع خطای فورانی با طول 32 یا کمتر، و خطاهای فورانی که تعداد بیتهای تغییر کرده فرد باشد، را آشکار می‌کند.

با اینکه بنظر می‌رسد محاسبهٔ مجموع چک و تست آن کار پیچیده‌ای باشد، پترسون و براون (1961) نشان دادند که می‌توان این کار را با یک مدار **شیفت رجیستر** (shift register) ساده بصورت سخت‌افزاری انجام داد. در واقع، این مدار در تمام کارتهای شبکه تعبیه شده است، و بسیاری از خطوط نقطه-به-نقطه هم از آن استفاده می‌کنند.

برای مدتهای مدید تصور بر آن بود که فریمهایی که مجموع چک آنها محاسبه می‌شود دارای طرح بیت تصادفی هستند، و تمام الگوریتمهای محاسبهٔ مجموع چک نیز فرض را بر این می‌گذاشتند. اما بررسی دقیق داده‌های واقعی نشان داده که این فرض بکلی اشتباه است. در نتیجه، خطاهایی که (تحت شرایط خاص) کشف نشده می‌مانند شایعتر از آن چیزیست که قبلاً تصور می‌شد (Partridge et al., 1995) .

۳-۳ چند پروتکل سادهٔ لینک داده

برای آشنایی با پروتکل‌های لایهٔ پیوند داده، در این قسمت سه پروتکل را (که بتدریج پیچیده‌تر می‌شوند) بررسی خواهیم کرد. برای خوانندگان علاقمند، شبیه‌ساز این پروتکلها (و پروتکلهایی که در آینده خواهید دید) را در سایت وب کتاب قرار داده‌ایم (http://www.prenhall.com/tanenbaum) . اما قبل از اینکه سراغ این پروتکلها برویم، اجازه دهید چند تا از فرض‌هایی را که دربارهٔ مدل ارتباطی زیربنایی داشته‌ایم، توضیح دهیم. اول اینکه فرض کرده‌ایم در لایه‌های فیزیکی، لایهٔ پیوند داده و شبکه پروسسهایی هستند که مستقل از یکدیگرند، و ارتباط آنها از طریق رد و بدل کردن پیام صورت می‌گیرد. در بسیاری از موارد، پروسسهای لایهٔ فیزیکی و لینک داده در پردازندهٔ کارت شبکه اجرا می‌شوند، و پروسسهای لایهٔ شبکه در CPU اصلی کامپیوتر. البته پیاده‌سازیهای دیگری نیز محتمل است (مثلاً، تمام پروسسهای لایهٔ فیزیکی، لینک داده و شبکه در پردازندهٔ کارت شبکه اجرا شوند، یا همگی آنها را CPU اصلی اجرا کند). در هر حال، مستقل دانستن این پروسسها بحث دربارهٔ آنها را بسیار ساده‌تر می‌کند، و تأکیدیست بر مستقل بودن لایه‌ها.

فرض کلیدی دیگر اینست که، ماشین A با استفاده از یک سرویس اتصال-گرای قابل اعتماد استریم طویلی از داده‌ها را به ماشین B می‌فرستد. بعدها این حالت که B هم همزمان به A داده بفرستد، را بررسی خواهیم کرد. علاوه بر آن، فرض کرده‌ایم ماشین A منبع بی‌پایانی از داده‌ها دارد که ارسال کند، و هرگز منتظر آمدن داده‌ها نخواهد شد. بعبارت دیگر، هر گاه لایهٔ پیوند داده A درخواست داده کند، لایهٔ شبکه بلافاصله اجابت می‌کند. (بعدها این فرض را هم کنار خواهیم گذاشت.)

فرض دیگر ما اینست که این کامپیوترها هرگز از کار نمی‌افتند؛ یعنی، پروتکلهای ما فقط با خطاهای مخابراتی سروکار دارند، نه هنگ کردن کامپیوتر و مسائلی از این قبیل.

دیگر اینکه، تا آنجا که به لایهٔ پیوند داده مربوط است، بسته‌ای که به لایهٔ شبکه داده می‌شود، **دادهٔ خالص** است، و باید تا آخرین بیت به آن تحویل شود. این که بخشی از این داده‌ها سرآیند بسته هستند و لایهٔ شبکه آنها را دور می‌ریزد، به خودش مربوط است نه به لایهٔ پیوند داده.

وقتی لایهٔ پیوند داده بسته‌ای از لایهٔ شبکه می‌گیرد تا ارسال کند، آنرا فریم‌بندی کرده و **سرآیند** (header) و **پی‌آیند** (trailer) های لازم را به آن می‌چسباند (شکل ۳-۱ را ببینید). بنابراین هر فریم از سه بخش تشکیل می‌شود: قسمتی از بسته‌ای که لایهٔ شبکه فرستاده، یک سرآیند شامل اطلاعات کنترلی، و یک پی‌آیند شامل مجموع چک فریم. سپس این فریم به لایهٔ پیوند داده ماشین مقصد فرستاده می‌شود. فرض ما بر این است که روال‌های کتابخانه‌ای *to_physical_layer* (برای ارسال) و *from_physical_layer* (برای دریافت) از قبل وجود دارند. مجموع چک نیز (مثلاً، با استفاده از کُدهای چندجمله‌ای) توسط سخت‌افزار محاسبه (و به انتهای فریم اضافه) می‌شود، بنابراین لازم نیست لایهٔ پیوند داده نگران آن باشد.

در ابتدا، لازم نیست گیرنده کاری انجام دهد؛ فقط منتظر می‌ماند تا اتفاقی بیفتد. در مثالهای این فصل، فرض کرده‌ایم که لایهٔ پیوند داده این کار را با روالی بنام *wait_for_event(&event)* انجام می‌دهد. این روال فقط وقتی به پایان می‌رسد (و کنترل را به برنامهٔ اصلی برمی‌گرداند) که اتفاقی افتاده باشد (یعنی، یک فریم دریافت شده باشد). اینکه چه اتفاقی افتاده است، را متغیر *event* مشخص می‌کند؛ و این که چه اتفاقهایی می‌تواند بیفتد، به تعریف پروتکل بستگی دارد. توجه داشته باشید که در دنیای واقعی لایهٔ پیوند داده (مانند این مثالها) در یک حلقهٔ بی‌انتها منتظر رسیدن فریمها نمی‌ماند، بلکه با استفاده از **وقفه** (interrupt) به آنها رسیدگی می‌کند. با این حال برای اجتناب از پیچیدگی مطلب، فرض کرده‌ایم که لایهٔ پیوند داده هیچ کار دیگری جز رسیدگی به کانال ارتباطی ما ندارد.

وقتی یک فریم به گیرنده می‌رسد، سخت‌افزار مجموع چک آنرا محاسبه می‌کند. اگر این مجموع چک اشتباه باشد (یعنی خطایی رخ داده)، به لایهٔ پیوند داده اطلاع داده می‌شود (*event = cksum_err*). اگر فریم بدرستی دریافت شده باشد، باز هم به اطلاع لایهٔ پیوند داده می‌رسد (*event = frame_arrival*). در این حالت لایهٔ پیوند داده با استفاده از تابع *from_physical_layer* فریم را گرفته، اطلاعات کنترلی موجود در سرآیند آنرا چک می‌کند، و اگر همه چیز مرتب باشد، سرآیند را جدا کرده و بخش اصلی داده را به لایهٔ شبکه تحویل می‌دهد.

تحت هیچ شرایطی سرآیند فریم به لایهٔ شبکه تحویل نمی‌شود، و برای این کار دلیل خوبی وجود دارد. پروتکلهای لینک داده و شبکه باید کاملاً از یکدیگر مستقل باشند. مستقل بودن پروتکلهای این دو لایه باعث می‌شود که بتوان هر کدام از این پروتکلها را تغییر داد، بدون اینکه نیاز باشد پروتکلهای لایهٔ دیگر تغییر کند (البته تحت هر شرایطی نحوهٔ تعامل و ارتباط لایه‌ها نباید تغییر کند). جدا و مستقل بودن لایه‌ها تا حد زیادی طراحی آنها را ساده می‌کند، چون می‌توان بدون نگرانی از اتفاقاتی که در لایه‌های دیگر می‌افتد، روی طراحی عملکردهای همان لایه تمرکز کرد.

در شکل ۳-۹ مقداری **تعریف** (به زبان C) می‌بینید، که در طراحی پروتکلهای این قسمت به آنها نیاز داریم. در اینجا پنج **ساختار داده** تعریف شده است: *boolean*، *seq_nr*، *packet*، *frame_kind* و *frame*. ساختار *boolean* از نوع **شمارشی** (enum) است، و می‌تواند دو مقدار *true* و *false* بگیرد. ساختار *seq_nr* از نوع **عدد صحیح بدون علامت** (unsigned int) تعریف شده، و برای شماره‌گذاری فریمها از آن استفاده خواهیم کرد شماره‌گذاری فریمها از 0 تا *MAX_SEQ* (که بسته به نیاز هر پروتکل تعریف می‌شود) انجام می‌گیرد. *packet* (بسته) واحدی از اطلاعات است که بین لایهٔ شبکه و لایهٔ پیوند داده (روی یک ماشین، یا روی ماشین‌های جداگانه) رد و بدل می‌شود. در مدل ما هر بسته همیشه حاوی *MAX_PKT* بایت داده است، ولی به واقعیت نزدیکتر است که طول بسته را متغیر در نظر بگیریم.

هر *frame* از چهار فیلد تشکیل شده: *kind*، *seq*، *ack* و *info* - که سه تای اول اطلاعات کنترلی هستند، و آخری همان داده‌هائیست که باید منتقل شود. به مجموعهٔ فیلدهای کنترلی سرآیند فریم (frame header) گفته می‌شود.

```
#define MAX_PKT 1024    /* determines packet size in bytes */

typedef enum {false, true} boolean;    /* boolean type */
typedef unsigned int seq_nr;    /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;    /*    packet definition */
typedef enum {data, ack, nak} frame_kind;    /* frame_kind definition */

typedef struct {    /* frames are transported in this layer */
  frame_kind kind;    /* what kind of a frame is it? */
  seq_nr seq;        /* sequence number */
  seq_nr ack;        /* acknowledgement number */
  packet info;       /* the network layer packet */
} frame;

/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
```

```
void  enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void  disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

شکل ۳-۹. تعریف‌های موردنیاز برای پروتکلهایی که در این فصل می‌نویسیم. این تعریف‌ها در فایلی بنام *protocol.h* قرار داده می‌شوند.

فیلد *kind* می‌گوید که آیا داده‌ای در فریم وجود دارد یا خیر، چون برخی از پروتکلها فریمهای بدون داده را از فریمهایی که داده دارند، تمیز می‌دهند. فیلدهای *seq* و *ack* بترتیب برای شمارهٔ ترتیبی فریم و تصدیق دریافت مورد استفاده قرار می‌گیرند؛ بعداً در این باره بیشتر توضیح خواهیم داد. دادهٔ اصلی (بسته) در فیلد *info* فریم قرار دارد؛ فریمهای کنترلی نیز وجود دارند که اساساً در آنها فیلد *info* وجود ندارد. در پروتکلهای واقعی که طول فیلد *info* می‌تواند متغیر باشد، نیازی به تمایز بین فریمهای داده و فریمهای کنترلی نیست (چون فریم کنترلی فریمی است که طول فیلد *info* در آن 0 است).

در اینجا لازم است تفاوت فریم (frame) و بسته (packet) را مجدداً یادآور شویم. لایهٔ شبکه با گرفتن پیام از لایه انتقال و اضافه کردن سرآیند، آنرا بصورت بسته در می‌آورد. سپس این بسته به لایهٔ پیوند داده تحویل می‌شود، که در آنجا در فیلد *info* یک فریم قرار داده شده و برای ارسال آماده می‌شود. وقتی این فریم به لایهٔ پیوند داده ماشین مقصد رسید، بسته از فیلد *info* استخراج شده و به لایهٔ شبکه تحویل می‌شود. این فرآیند بکلی شفاف است، و لایه‌های شبکه در دو ماشین متقابل تصور می‌کنند که مستقیماً در حال تبادل بسته‌اند.

در برنامهٔ شکل ۳-۹ چند روال (تابع) نیز تعریف شده است. اینها روالهای کتابخانه‌ای هستند، که فقط کاری که انجام می‌دهند برای ما مهم است (و اصلاً اهمیتی ندارد این کار را چگونه انجام می‌دهند). روال *wait_for_event* (همانطور که قبلاً گفتیم) در یک حلقهٔ بی‌انتها به انتظار می‌ماند تا اتفاقی بیفتد. روالهای *to_network_layer* و *from_network_layer* بترتیب برای ارسال بسته به لایهٔ شبکه و برای گرفتن بسته از این لایه بکار می‌روند (اینها واسط لایه‌های ۲ و ۳ هستند). برای تبادل اطلاعات با لایهٔ فیزیکی نیز از روالهای *to_physical_layer* و *from_physical_layer* استفاده می‌شود (اینها واسط لایه‌های ۱ و ۲ هستند).

در پروتکلهای واقعی فرض بر اینست که کانال ارتباطی نامطمئن است، و احتمال این هست که فریمها در راه از بین بروند. برای مقابله با چنین وضعیتی، لایهٔ پیوند داده همزمان با ارسال هر فریم، باید یک تایمر را راه‌اندازی کند. اگر بعد از مدتی معین پاسخی از طرف مقابل نرسید، تایمر مزبور لایهٔ پیوند داده را (با استفاده از یک وقفه) مطلع می‌کند.

در پروتکلهای ما، در چنین وضعیتی روال *wait_for_event* مقدار *event = timeout* برمی‌گرداند. روالهای *start_timer* و *stop_timer* نیز بترتیب تایمر را روشن و خاموش می‌کنند (البته سپری شدن زمان مقتضی - *timeout* - فقط زمانی اتفاق می‌افتد که تایمر روشن باشد). یک تایمر را می‌توان (قبل از منقضی شدن آن) با اجرای مجدد روال *start_timer* ریست کرد.

روالهای *start_ack_timer* و *stop_ack_timer* تایمر دیگری را (برای ایجاد فریمهای تصدیق دریافت در شرایطی خاص) کنترل می‌کنند.

از روالهای *enable_network_layer* و *disable_network_layer* در پروتکلهای پیچیده‌تر استفاده می‌شود (ما در پروتکلهای سادهٔ این قسمت از این روالها استفاده نخواهیم کرد، چون فرض کرده‌ایم که لایهٔ شبکه همیشه می‌تواند به لایهٔ پیوند داده بسته تحویل دهد). وقتی لایهٔ پیوند داده لایهٔ شبکه را فعال می‌کند (*enable_network_layer*)، لایهٔ شبکه اجازه دارد آماده شدن بستهٔ داده را با یک وقفه به لایهٔ پیوند داده اطلاع دهد (این کار با *event = network_layer_ready* انجام می‌شود). اگر لایهٔ شبکه غیرفعال باشد (*disable_network_layer*)، اجازهٔ چنین کاری را ندارد. با استفاده دقیق و بجا از روالهای *enable_network_layer* و *disable_network_layer*، لایهٔ پیوند داده می‌تواند مطمئن شود که (هنگام پر شدن بافر) در سیلاب بسته‌های ارسالی از لایهٔ شبکه غرق نخواهد شد.

شمارهٔ ترتیبی فریم همیشه بین 0 تا *MAX_SEQ* (و از جمله خود این دو عدد) است، که البته *MAX_SEQ* در پروتکلهای مختلف می‌تواند متفاوت باشد. شمارهٔ ترتیبی فریمها معمولاً یکی یکی اضافه می‌شود، و وقتی به *MAX_SEQ* رسید، دوباره 0 خواهد شد - این کار بر عهدهٔ ماکروی *inc* گذاشته شده است. برای سرعت بخشیدن به اجرای این عملیات، *inc* بصورت **ماکرو** (macro) تعریف شده است. [کامپایلر با دیدن یک ماکرو، دستور معادل را جایگزین آن می‌کند، و مانند تابع آنرا فراخوانی نمی‌کند. م.] همانطور که بعداً خواهید دید، سرعت اجرای پروتکلها یکی از عوامل کلیدی در کارایی شبکه است، و استفاده از ماکرو (بجای تابع) تأثیر زیادی بر بهبود این کارایی دارد. همچنین، از آنجائیکه *MAX_SEQ* در پروتکلهای مختلف مقادیر متفاوتی دارد، تعریف *inc* بصورت ماکرو امکان می‌دهد تا بدون هیچ مشکلی از آن در پروتکلهای مختلف استفاده کنیم.

تعاریف شکل ۳-۹ بخشی از پروتکلهاییست که در قسمتهای آینده خواهیم نوشت. البته می‌توانستیم آنها را در ابتدای هر پروتکل نیز بیاوریم، ولی با جمع کردن آنها در یک فایل کُدهای آینده بسیار ساده‌تر خواهند شد. در زبان C، این کار با استفاده از دستور include# و نوشتن نام فایل تعاریف (در اینجا *protocol.h*) انجام می‌شود.

۳-۳-۱ پروتکل یکطرفهٔ نامقید

در اولین مثال یک پروتکل بسیار ساده را در نظر می‌گیریم، که در آن داده‌ها فقط در یک جهت منتقل می‌شوند. لایهٔ شبکه در فرستنده و گیرنده آمادهٔ کار هستند، زمان پردازش را می‌توان نادیده گرفت، از نظر بافر هیچ کمبودی وجود ندارد، و مهمتر از همه اینکه کانال ارتباطی بین دو لایهٔ پیوند داده کامل و بدون نقص است، و هیچ خطایی در آن رخ نمی‌دهد. این پروتکل غیرواقعی (که شاید «**اتوپیا**» - پروتکل آرمانی - مناسبترین نام برای آن باشد) را در شکل ۳-۱۰ ملاحظه می‌کنید.

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver.  The communication channel is assumed to be error free
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
```

```
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */

  while (true) {
        from_network_layer(&buffer);     /* go get something to send */
        s.info = buffer;                 /* copy it into s for transmission */
        to_physical_layer(&s);           /* send it on its way */
  }                                     /* Tomorrow, and tomorrow, and tomorrow,
                                           Creeps in this petty pace from day to day
                                           To the last syllable of recorded time.
                                              - Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;                      /* filled in by wait, but not used here */

  while (true) {
        wait_for_event(&event);          /* only possibility is frame_arrival */
        from_physical_layer(&r);         /* go get the inbound frame */
        to_network_layer(&r.info);       /* pass the data to the network layer */
  }
}
```

شکل ۳-۱۰. پروتکل یکطرفهٔ نامقید.

این پروتکل دارای دو روال مجزا است: فرستنده و گیرنده. فرستنده در لایهٔ پیوند داده ماشین مبدأ، و گیرنده در لایهٔ پیوند داده ماشین مقصد اجرا می‌شود. در اینجا از شمارهٔ ترتیبی فریمها و تصدیق دریافت استفاده‌ای نمی‌شود، بنابراین به *MAX_SEQ* هم نیازی نیست. تنها رویداد قابل انتظار *frame_arrival* (رسیدن صحیح و سالم فریم) است.

فرستنده یک حلقهٔ بی‌انتهای while است که داده‌ها را با حداکثر توان به بیرون پمپ می‌کند. بدنهٔ این حلقه سه کار انجام می‌دهد: آوردن یک بسته از لایهٔ شبکه (که همیشه آمادهٔ خدمت است)، ایجاد فریم خروجی با استفاده از متغیر *s*، و فرستادن فریم. این پروتکل فقط از فیلد *info* فریم استفاده می‌کند، چون فیلدهای دیگر مربوط به کنترل جریان و خطا هستند، که طبق فرض ما چنین محدودیتهایی در اینجا وجود ندارد.

گیرنده هم بهمان اندازه ساده است: انتظار بی‌پایان برای دریافت فریمی که همیشه سالم و بی‌نقص است. وقتی یک فریم از راه رسید، روال *wait_for_event* کنترل را به برنامهٔ اصلی برمی‌گرداند، و متغیر *event* را به *frame_arrival* ست می‌کند (که بهر حال استفاده‌ای از آن نمی‌شود). با فراخوانی روال *from_physical_layer*، فریم تازه از راه رسیده از بافر سخت‌افزاری برداشته شده و در متغیر *r* قرار داده می‌شود (تا گیرنده می‌تواند آنرا بردارد). در پایان، بخش دادهٔ این فریم (فیلد *info*) به لایهٔ شبکه فرستاده شده، و لایهٔ پیوند داده به انتظار فریم بعدی می‌نشیند.

۳-۳-۲ پروتکل توقف-انتظار یکطرفه

حال اجازه دهید غیرواقعی‌ترین بخش از پروتکل ۱ را کنار بگذاریم: نامحدود بودن توانایی این پروتکل در دریافت بسته‌ها از لایهٔ شبکه، و پردازش فریمهای ورودی (که به معنای نامحدود بودن بافر لایهٔ پیوند داده در سمت گیرنده است). اما کانال ارتباطی را همچنان بدون خطا، و ارتباط را یکطرفه فرض کرده‌ایم.

مهمترین مشکلی که با آن روبرو هستیم، این است که چگونه از غرق شدن گیرنده در سیلاب فریمهایی (که پردازش آنها از توان وی خارج است) جلوگیری کنیم. واضحست که، اگر گیرنده برای اجرای روالهای *from_physical_layer* و *to_network_layer* به زمان Δt نیاز داشته باشد، سرعت متوسط ارسال فرستنده بایستی از یک فریم بر Δt کمتر باشد. همچنین اگر فرض کنیم که سخت‌افزار گیرنده بطور خودکار عمل بافر کردن فریمها را انجام نمی‌دهد، فرستنده نباید قبل از برداشته شدن یک فریم از بافر لایهٔ فیزیکی (که توسط روال *from_physical_layer* انجام می‌شود)، فریم بعدی را ارسال کند چون در غیر اینصورت فریم قبلی از بین می‌رود (به این حالت **روهم‌نویسی** - overrun - می‌گویند).

در شرایط خاصی (مانند ارتباط سنکرون، و گیرنده‌ای که تنها وظیفهٔ آن گرفتن اطلاعات از خط ورودی است)، با ایجاد تأخیر در قسمت فرستندهٔ پروتکل ۱ و کُند کردن آن می‌توان به اهداف فوق دست یافت. اما بسیار محتملتر است که یک لایهٔ پیوند داده مجبور باشد چندین خط ورودی را پردازش کند، که در این حالت فاصلهٔ زمانی دریافت فریمها و پردازش آنها می‌تواند بسیار متغیر باشد. اگر طراحان شبکه بتوانند بدترین حالت گیرنده را محاسبه کنند، می‌توانند فرستنده را آنقدر کُند کنند که روهم‌نویسی هرگز اتفاق نیفتد. اما این روش بسیار محافظه‌کارانه است و بنحو بسیار بدی پهنای باند را تلف می‌کند، مگر اینکه تفاوت بهترین و بدترین حالت چندان زیاد نباشد (یعنی تفاوت پاسخهای لایهٔ پیوند داده ناچیز باشد).

راه حل بهتر این معضل، برگرداندن **بازخور** (feedback) از گیرنده به فرستنده است. بعد از تحویل بسته به لایهٔ شبکه، گیرنده یک فریم کوچک (که لازم نیست معنی خاصی هم داشته باشد) به فرستنده می‌فرستد، که در واقع مجوز ارسال فریم بعدی محسوب می‌شود. فرستنده بعد از ارسال یک فریم، آنقدر منتظر می‌ماند تا این فریم کوچک (که در واقع همان **تصدیق دریافت** - acknowledgement - است) از راه برسد. استفاده از بازخور گیرنده برای اطلاع به فرستنده (و دادن مجوز ارسال فریمهای بعدی) یکی از نمونه‌های **کنترل جریان** (flow control)، که قبلاً به آن اشاره کردیم، است.

پروتکلهایی که در آنها فرستنده قبل از ارسال فریم بعدی منتظر تصدیق دریافت فریم قبلی از گیرنده می‌ماند، به پروتکلهای **توقف-انتظار** (stop-and-wait) معروفند. در شکل ۳-۱۱ یک نمو از پروتکلهای توقف-انتظار را ملاحظه می‌کنید.

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
"h.locotorp" edulcni#
```

```
void sender2(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;                     /* frame_arrival is the only possibility */

  while (true) {
        from_network_layer(&buffer);    /* go get something to send */
        s.info = buffer;                /* copy it into s for transmission */
        to_physical_layer(&s);          /* bye-bye little frame */
        wait_for_event(&event);         /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers for frames */
  event_type event;                     /* frame_arrival is the only possibility */
  while (true) {
        wait_for_event(&event);         /* only possibility is frame_arrival */
        from_physical_layer(&r);        /* go get the inbound frame */
        to_network_layer(&r.info);      /* pass the data to the network layer */
        to_physical_layer(&s);          /* send a dummy frame to awaken sender */
  }
}
```

شکل ۳-۱۱. پروتکل توقف-انتظار یکطرفه.

با اینکه این پروتکل **یکطرفه** (simplex) است (یعنی ما فقط در یک جهت ارسال می‌کنیم)، اما فریمها می‌توانند در هر دو جهت رفت و آمد کنند. برای این منظور لازم است کانال ارتباطی ما چنین قابلیتی داشته باشد؛ البته یک کانال **دوطرفهٔ ناهمزمان** (half-duplex) هم کفایت می‌کند، چون فرستنده و گیرنده در آن واحد اقدام به فرستادن فریمها نمی‌کنند: فرستنده یک فریم می‌فرستد، گیرنده پاسخ می‌دهد، فرستنده فریم بعدی را می‌فرستد، گیرنده پاسخ می‌دهد، و الی آخر.

در اینجا هم (مانند پروتکل ۱) فرستنده همان سه کار قبلی را انجام می‌دهد: آوردن یک بسته از لایهٔ شبکه، ایجاد فریم خروجی، و فرستادن آن. اما برخلاف پروتکل ۱، قبل از ادامهٔ کار (آوردن بستهٔ بعدی و ارسال آن در قالب یک فریم) باید منتظر رسیدن فریم تصدیق دریافت از گیرنده بماند. نیازی نیست که لایهٔ پیوند داده فرستنده فریم دریافتی را بررسی کند، چون فقط یک احتمال وجود دارد: این فریم همیشه تصدیق دریافت گیرنده است.

تنها تفاوت *receiver2* با *receiver1* این است که بعد از تحویل بسته به لایهٔ شبکه و قبل از ورود به حالت انتظار، *receiver2* یک فریم تصدیق دریافت به فرستنده باز پس می‌فرستد. از آنجائیکه فقط خود فریم مهم است و نه محتویات آن، گیرنده هیچ داده‌ای در فیلد *info* این فریم قرار نمی‌دهد.

۳-۳-۳ پروتکل یکطرفه برای کانال‌های نویزدار

اکنون به یک حالت واقعی‌تر می‌پردازیم: کانالهایی که نویز دارند. فریمها می‌توانند با خطا به مقصد برسند، و یا بکلی گم شده و اصلاً به مقصد نرسند. با این حال، فرض می‌کنیم که اگر فریمی با خطا به مقصد رسید، سخت‌افزار لایهٔ فیزیکی با محاسبهٔ جمع تطبیقی متوجه خطا می‌شود. پروتکل ما در یک حالت به اشتباه عمل خواهد کرد: خطای رخ داده آنقدر شدید باشد که جمع تطبیقی تصادفاً درست از کار در آید (اتفاقی که بسیار نامحتمل است).

در نگاه اول با یک تغییر کوچک در پروتکل ۲ (اضافه کردن یک تایمر) می‌توان آن را با وضعیت جدید تطبیق داد. فرستنده می‌تواند در هر زمانی یک فریم بفرستد، ولی گیرنده فقط وقتی فریم تصدیق دریافت را برمی‌گرداند که این فریم را بدرستی دریافت و پردازش کرده باشد. اگر فریم ناقص به مقصد برسد، دور انداخته خواهد شد. بعد از مدتی تایمر فرستنده به انتها می‌رسد، و چون هنوز تصدیق دریافت گیرنده را نگرفته، مجدداً اقدام به ارسال فریم می‌کند. این ماجرا تا زمانی که فریم به سلامت به مقصد برسد، تکرار خواهد شد.

اما طرح بالا یک مشکل اساسی دارد. قبل از خواندن ادامهٔ کتاب، کمی فکر کنید و ببینید می‌توانید متوجه اشکال آن شوید.

برای درک مشکل، بیاد بیاورید که وظیفهٔ برقراری یک کانال ارتباطی عاری از خطا بین لایه‌های شبکه بر عهدهٔ لایهٔ پیوند داده است. لایهٔ شبکهٔ ماشین A یک سری بسته به لایهٔ پیوند داده می‌دهد، و باید مطمئن باشد که این بسته‌ها به همان ترتیبی که ارسال شده‌اند بدست لایهٔ شبکهٔ ماشین B خواهند رسید. بویژه، لایهٔ شبکهٔ ماشین B هیچ راهی ندارد تا بفهمد که یک بسته گم شده یا تکراریست. به همین دلیل لایهٔ پیوند داده ماشین B باید تضمین کند که هیچ بسته‌ای گم نمی‌شود، و یا تکراری نیست.

سناریوی زیر را در نظر بگیرید:

۱. لایهٔ شبکهٔ ماشین A بستهٔ ۱ را به لایهٔ پیوند داده می‌دهد. این بسته صحیح و سالم به لایهٔ پیوند داده ماشین B می‌رسد، و تحویل لایهٔ شبکه می‌شود. ماشین B یک فریم تصدیق دریافت به A می‌فرستد.
۲. فریم تصدیق دریافت ماشین B در راه از بین می‌رود، و هرگز به A نمی‌رسد. اگر فقط فریمهای داده گم می‌شدند و این اتفاق برای فریمهای کنترلی نمی‌افتاد، زندگی چقدر شیرین‌تر بود! ولی متأسفانه کانالهای مخابراتی اهل تبعیض نیستند!
۳. تایمر لایهٔ پیوند داده A به انتها می‌رسد، و چون هیچ فریم تصدیق دریافتی بدستش نرسیده، (باشتباه) تصور می‌کند که فریم به مقصد نرسیده (یا خراب شده)، پس آنرا دوباره می‌فرستد.
۴. فریم تکراری (در کمال صحت و سلامت) به لایهٔ پیوند داده B می‌رسد، و این لایه هم (بی خبر از همه جا) آنرا به لایهٔ شبکه تحویل می‌دهد. تصور کنید که اگر A در حال ارسال یک فایل به B باشد، تکراری بودن بخشی از آن چه فاجعه‌ای ببار خواهد آورد. همانطور که می‌بینید، پروتکل ما یک شکست کامل است.

چیزی که ما به آن احتیاج داریم، وسیله‌ایست که بتوان فریمهای تکراری را از فریمهایی که برای اولین بار دریافت می‌شوند، تشخیص داد. راه حل واضح این مشکل آن است که فرستنده یک **شمارهٔ ترتیبی** (sequence number) در سرآیند فریمهایی که می‌فرستد، قرار دهد. گیرنده می‌تواند این شماره را چک کرده، و فریمهای تکراری را دور بیندازد.

از آنجائیکه سرآیند یک فریم باید حتی‌الامکان کوچک باشد، سؤالی که پیش می‌آید اینست که: «حداقل تعداد بیتهای لازم برای فیلد شمارهٔ ترتیبی چندتاست؟» در پروتکل ما تنها ابهام در فریم m و فریم بعدی آن یعنی $m + 1$ است. اگر فریم m خراب شود یا از بین برود، گیرنده فریم تصدیق دریافت آنرا برنمی‌گرداند، پس فرستنده سعی می‌کند آنرا دوباره بفرستد. همین که این فریم سالم به مقصد رسید، گیرنده فریم تصدیق دریافت را به فرستنده پس می‌فرستد. همین جاست که مشکل بروز می‌کند: اگر فریم تصدیق دریافت صحیح و سالم به فرستنده بفرستد،

فرستنده فریم بعدی (یعنی $m + 1$) را می‌فرستد، در غیر اینصورت فریم m را خواهد فرستاد.

برای ارسال فریم $m + 2$، فرستنده باید قبلاً تصدیق دریافت فریم $m + 1$ را گرفته باشد. اما این بدان معناست که فریم m به سلامت به مقصد رسیده و تصدیق دریافت آن هم بدرستی به فرستنده برگشت داده شده است (چون در غیر اینصورت فرستنده فریم $m + 1$ را هم نمی‌فرستاد، چه رسد به فریم $m + 2$).

بنابراین، تنها ابهامی که می‌تواند وجود داشته باشد، بین یک فریم و فریم بعدی آن است. برای تشخیص این دو هم یک شمارهٔ ترتیبی یک‌بیتی (0 یا 1) کافیست. بعبارت دیگر، در هر لحظه گیرنده باید بدنبال شمارهٔ بعدی باشد. اگر فریمی با شمارهٔ اشتباه دریافت شد، گیرنده آنرا تکراری تلقی کرده و دور می‌اندازد. اما اگر شمارهٔ ترتیبی فریم درست بود، به لایهٔ شبکه تحویل داده می‌شود. با این توصیف فیلد شمارهٔ ترتیبی باید در مدول 2 افزایش داده شود (بعبارت دیگر، 1 به 0 تبدیل می‌شود، و 0 به 1).

پروتکلی با این مشخصات را در شکل ۳-۱۲ ملاحظه می‌کنید. به پروتکلهایی که فرستنده برای ارسال فریم بایستی منتظر یک تصدیق دریافت مثبت بماند، PAR (**تصدیق دریافت مثبت با ارسال مجدد** – Positive Acknowledgement with Retransmission) یا ARQ (**درخواست تکرار خودکار** – Automatic Repeat reQuest) نیز گفته می‌شود. این پروتکل هم، مانند پروتکل ۲، فقط در یک جهت داده می‌فرستد.

تفاوت پروتکل ۳ با دو تای قبلی اینست که، روالهای فرستنده و گیرنده متغیری دارند که مقدار آن حتی در زمانی که لایهٔ پیوند داده به حالت انتظار می‌رود، دست نخورده باقی می‌ماند. فرستنده باید شمارهٔ ترتیبی فریم بعدی که می‌خواهد بفرستد، را بداند: *next_frame_to_send*؛ و گیرنده هم باید شمارهٔ ترتیبی فریم بعدی که باید منتظر آن باشد، را بداند: *frame_expected*.

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */
#define MAX_SEQ 1                          /* must be 1 for protocol 3 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void sender3(void)
{
  seq_nr  next_frame_to_send;              /* seq number of next outgoing frame
*/
   frame s;                               /* scratch variable */
   packet buffer;                         /* buffer for an outbound packet */
   event_type  event;
   next_frame_to_send = 0;                /*   initialize   outbound   sequence
numbers */
   from_network_layer(&buffer);           /* fetch first packet */
   while (true) {
        s.info = buffer;                  /* construct a frame for transmission
*/
        s.seq = next_frame_to_send;       /* insert sequence number in frame
*/
        to_physical_layer(&s);            /* send it on its way */
        start_timer(s.seq);               /* if answer takes too long, time out
```

```
*/
            wait_for_event(&event);                /* frame_arrival, cksum_err, timeout
*/
            if (event == frame_arrival) {
                  from_physical_layer(&s);         /* get the acknowledgement */
                  if (s.ack == next_frame_to_send) {
                        stop_timer(s.ack);         /* turn the timer off */
                        from_network_layer(&buffer); /* get the next one to send */
                        inc(next_frame_to_send);   /* invert next_frame_to_send */
                  }
            }
      }
   }

   void receiver3(void)
   {
      seq_nr frame_expected;
      frame r, s;
      event_type event;
      frame_expected = 0;
      while (true) {
            wait_for_event(&event);                /*     possibilities:     frame_arrival,
cksum_err */
            if (event == frame_arrival) {          /* a valid frame has arrived. */
                  from_physical_layer(&r);         /* go get the newly arrived frame
*/
                  if (r.seq == frame_expected) {   /*  this  is  what  we  have  been
waiting for. */
                        to_network_layer(&r.info); /* pass the data to the network
layer */
                        inc(frame_expected);       /*  next  time  expect  the  other
sequence nr */
                  }
                  s.ack = 1 - frame_expected;      /* tell which frame is being acked
*/
                  to_physical_layer(&s);           /* send acknowledgement */
            }
      }
   }
```

شکل ۳-۱۲. پروتکل تصدیق دریافت مثبت با ارسال مجدد (PAR) .

فرستنده، بعد از ارسال فریم، تایمر را راه می‌اندازد. اگر تایمر از قبل در حال اجرا باشد، این کار آنرا ریست کرده و آمادهٔ کار بعدی می‌کند. فاصلهٔ زمانی تایمر باید بگونه‌ای انتخاب شود که وقت کافی برای سه رویداد بدست دهد: رسیدن فریم به گیرنده، پردازش آن در گیرنده (در بدترین حالت)، و برگشت فریم تصدیق دریافت به فرستنده. فقط پس از سپری شدن این زمان است که می‌توان مطمئن شد فریم (یا تصدیق دریافت آن) به مقصد نرسیده، و فرستنده باید دوباره آنرا بفرستد. اگر فاصلهٔ زمانی تایمر کم انتخاب شود، تعداد دفعاتی که فرستنده فریم تکراری می‌فرستد افزایش یافته، و (با اینکه این کار تأثیر منفی روی گیرنده ندارد) به کارایی سیستم لطمه می‌زند.

بعد از ارسال فریم و راه انداختن تایمر، فرستنده منتظر یک اتفاق هیجان‌انگیز می‌ماند. البته فقط سه احتمال برای چنین اتفاقی وجود دارد: فریم تصدیق دریافت صحیح و سالم از راه برسد، فریم تصدیق دریافت با خطا وارد شود، تایمر منقضی شود (زمان مشخص شده به انتها برسد). در حالت اول، فرستنده بستهٔ دیگری از لایهٔ شبکه گرفته، و در بافر خود (*buffer*) قرار می‌دهد. سپس، شمارهٔ ترتیبی فریم (*next_frame_to_send*) را بالا می‌برد. اما در دو حالت دیگر (خراب شدن فریم تصدیق دریافت یا نرسیدن آن)، بافر و شمارهٔ ترتیبی هیچکدام تغییری نمی‌کند، بنابراین در هیچ حالتی فریم تکراری فرستاده نخواهد شد.

وقتی یک فریم به گیرنده می‌رسد، گیرنده شمارهٔ ترتیبی آن را چک می‌کند، تا از تکراری نبودن آن مطمئن شود. این فریم فقط در صورت تکراری نبودن به لایهٔ شبکه تحویل داده می‌شود. بدین ترتیب، فریمهای تکراری و خراب به لایهٔ شبکه نخواهند رسید.

۴ـ۳ پروتکل‌های پنجرهٔ لغزنده

در پروتکلهای قبل، فریمهای داده فقط در یک جهت ارسال می‌شدند. اما در عمل باید بتوانیم در هر دو جهت انتقال داده داشته باشیم. یکی از راههای داشتن یک کانال **دوطرفهٔ همزمان** (full-duplex) استفاده از دو کانال **یکطرفه** (simplex) در دو جهت مخالف است، که هر کدام فقط در یک جهت داده می‌فرستند (و البته فریمهای تصدیق دریافت می‌کنند). اما این روش چیزی جز اتلاف پهنای باند (و پول) نیست.

روش بهتر استفاده از یک کانال واحد برای ارسال داده در دو جهت است (مگر نه اینکه در پروتکلهای ۲ و ۳ در هر دو جهت فریم ارسال کردیم). از آنجائیکه در این مدل، فریمهای داده و تصدیق دریافت در هر دو جهت می‌توانند فرستاده شوند، باید کاری کنیم که گیرنده بتواند آنها را از یکدیگر تشخیص دهد. برای این منظور می‌توانیم از فیلد *info* در سرآیند فریمها استفاده کنیم.

با اینکه ترکیب فریمهای داده و تصدیق دریافت روی یک مدار واحد (بجای دو مدار جداگانه) یک قدم به جلو محسوب می‌شود، اما باز هم می‌توان کارایی سیستم را بهبود بخشید. وقتی گیرنده یک فریم داده دریافت می‌کند، بجای اینکه بلافاصله یک فرم کنترلی پس بفرستد، منتظر می‌ماند تا بستهٔ بعدی را برای ارسال از لایهٔ شبکه بگیرد. تصدیق دریافت فریم قبلی در فیلد *ack* فریم داده‌ای که اکنون می‌خواهد فرستاده شود، قرار داده می‌شود، و در واقع فریم تصدیق دریافت از فریم داده **سواری‌مجانی** (piggyback) می‌گیرد (و به همین نام هم خوانده می‌شود).

یکی از مزایای تکنیک سواری مجانی نسبت به ارسال مستقل فریمهای تصدیق دریافت، استفاده بهینه‌تر از پهنای باند موجود است: فیلد *ack* فقط چند بیت از سرآیند را اشغال می‌کند، در حالیکه یک فریم مستقل برای خود سرآیند، جمع تطبیقی، و تصدیق دریافت دارد. علاوه بر آن، هر چه تعداد فریمهایی که در یک جهت فرستاده می‌شوند کمتر باشد، گیرنده بهتر می‌تواند به کارهای دیگرش (پردازش فریمهای رسیده و خالی کردن بافرها) برسد، و این هم به بهبود کارایی سیستم کمک می‌کند. در پروتکلی که در این قسمت می‌نویسیم، فیلد سواری مجانی فقط یک بیت به سرآیند فریم اضافه می‌کند (و بندرت پیش می‌آید که مقدار آن از چند بیت بیشتر شود).

اما سواری مجانی هم خالی از اشکال نیست. برای مثال، در این حالت لایهٔ پیوند داده گیرنده چقدر باید منتظر بسته از لایهٔ شبکهٔ خود شود؟ اگر خیلی منتظر بماند، تایمر فرستنده به انتها رسیده و فریم را تکرار می‌کند، که این نقض غرض (از ارسال فریم تصدیق دریافت) است. اگر لایهٔ پیوند داده قدرت پیشگویی داشت، می‌توانست زمان دریافت بستهٔ بعدی از لایهٔ شبکه را پیشگویی کند، و آنوقت می‌توانست تصمیم بگیرد که منتظر این بسته بماند یا بلافاصله فریم تصدیق دریافت را به فرستنده بفرستد. اما متأسفانه لایهٔ پیوند داده نمی‌تواند آینده را پیشگویی کند، پس باید روش ساده‌تری (انتظار بمدت ثابت، مثلاً چند میلی‌ثانیه) پیدا کنیم. اگر در این فاصله بسته‌ای از لایهٔ شبکه رسید، لایهٔ پیوند داده تصدیق دریافت را سوار آن می‌کند؛ در غیر اینصورت یک فریم مستقل تصدیق دریافت به سمت فرستنده ارسال می‌کند.

پروتکلهایی که در این قسمت خواهید دید، به کلاس پروتکلهای **پنجرهٔ لغزنده** (sliding window) تعلق دارند، که فقط از نظر کارایی، پیچیدگی و بافر با هم تفاوت دارند. در این پروتکلها، مانند سایر پروتکلهای پنجرهٔ لغزنده، هر فریم خروجی یک شمارهٔ ترتیبی (از 0 تا یک حداکثر) دارد. حداکثر شمارهٔ فریمها معمولاً $2^n - 1$ است، بنابراین در یک فیلد n-بیتی بخوبی جا می‌شود. پروتکل پنجرهٔ لغزندهٔ توقف-انتظار از $n = 1$ استفاده می‌کرد، اما در پروتکلهای دیگر این عدد می‌تواند بیشتر باشد.

ایدهٔ اصلی در تمام پروتکلهای پنجرهٔ لغزنده این است که، فرستنده در هر لحظه از زمان لیستی از شماره‌های ترتیبی متناظر با فریمهایی که می‌تواند ارسال کند، در اختیار دارد. اصطلاحاً گفته می‌شود که این فریمها در **پنجرهٔ ارسال** (sending window) قرار دارند. گیرنده هم یک **پنجرهٔ دریافت** (receiving window) دارد که متناظر است با فریمهایی که مجاز به دریافت آنهاست. الزامی نیست که پنجرهٔ ارسال و پنجرهٔ دریافت حد پائین و بالای مشابه داشته باشند، یا حتی هم‌اندازه باشند. در برخی پروتکلها اندازهٔ این پنجره‌ها ثابت است، ولی در پروتکلهای دیگر می‌تواند کوچک یا بزرگ شود.

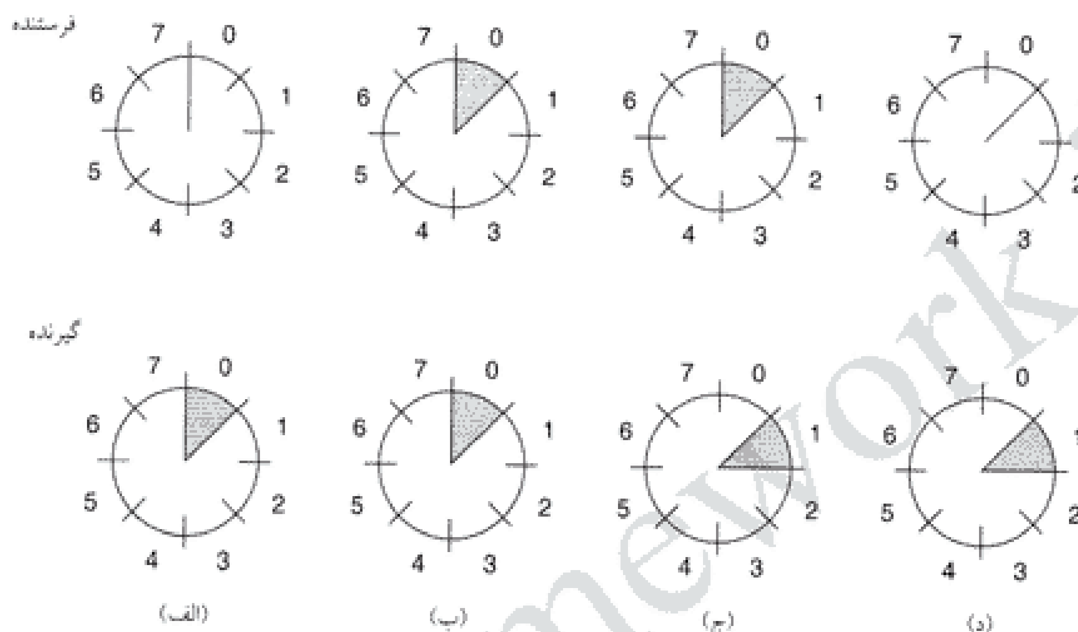
با اینکه این پروتکلها آزادی عمل بیشتری به لایهٔ پیوند داده در ارسال و دریافت فریمها می‌دهند، لازم است مجدداً تأکید کنیم که تحویل بسته‌ها به لایهٔ شبکهٔ مقصد باید با همان ترتیبی صورت گیرد که در ماشین مبدأ تحویل لایهٔ پیوند داده شده‌اند. (و یک بار دیگر خاطرنشان می‌کنیم که، لایهٔ فیزیکی یک کانال ارتباطی ساده است که فریمها را به همان ترتیبی که به آن داده شده، منتقل می‌کند.)

شماره‌های موجود در پنجرهٔ ارسال شمارهٔ فریمهائیست که باید ارسال شوند، و یا ارسال شده‌اند ولی هنوز تصدیق دریافت آنها برنگشته است. وقتی یک بستهٔ جدید از لایهٔ شبکه می‌رسد، لایهٔ پیوند داده بالاترین شمارهٔ ترتیبی موجود را به آن می‌دهد، و لبهٔ بالایی پنجرهٔ ارسال را یکی زیاد می‌کند. وقتی یک تصدیق دریافت وارد شد،لبهٔ پائینی پنجرهٔ ارسال نیز بالا برده می‌شود. بدین ترتیب پنجرهٔ ارسال همیشه شامل لیست فریمهائیست که دریافت آنها هنوز توسط گیرنده تصدیق نشده است. در شکل ۳-۱۳ یک نمونه را ملاحظه می‌کنید.

از آنجائیکه امکان گم یا خراب شدن فریمهایی که در حال حاضر در پنجرهٔ ارسال قرار دارند، همیشه وجود دارد، فرستنده باید آنها را برای ارسال مجدد (احتمالی) در حافظه نگه دارد. بنابراین اگر حداکثر اندازهٔ این پنجره n باشد، فرستنده باید بافری باندازهٔ n فریم تصدیق نشده داشته باشد. اگر پنجرهٔ ارسال از حداکثر پیش‌بینی شده بزرگتر شود، لایهٔ پیوند داده باید گرفتن بسته از لایهٔ شبکه را تا زمان آزاد شدن بافر متوقف کند.

پنجرهٔ دریافت در گیرنده متناظر است با فریمهایی که گیرنده مجاز به دریافت آنهاست. هر فریمی که خارج از این پنجره قرار گیرد، بدون هیچ توضیحی دور انداخته خواهد شد. وقتی فریمی که شمارهٔ ترتیبی آن معادل لبهٔ پائین پنجرهٔ دریافت است، از راه می‌رسد، به لایهٔ شبکه تحویل شده و پس از ایجاد تصدیق دریافت آن، پنجرهٔ دریافت یک واحد می‌چرخد. بر خلاف پنجرهٔ فرستنده، پنجرهٔ گیرنده همیشه به همان اندازهٔ اولیه می‌ماند. توجه

کنید که پنجرهٔ دریافت 1 بمعنای اینست که لایهٔ پیوند داده فریمها را فقط به ترتیب می‌پذیرد، ولی در پنجره‌های بزرگتر الزاماً چنین نیست (با این حال، لایهٔ شبکه همیشه داده‌ها را به ترتیب صحیح تحویل لایهٔ پیوند داده می‌دهد).



شکل ۳-۱۳. یک پنجرهٔ لغزندهٔ یک واحدی، با شمارهٔ ترتیبی ۳-بیتی. (الف) در شروع کار. (ب) بعد از ارسال اولین فریم. (ج) بعد از آنکه اولین فریم دریافت شد. (د) بعد از آنکه فرستنده اولین تصدیق دریافت را گرفت.

در شکل ۳-۱۳ حداکثر اندازهٔ پنجره 1 است. در لحظهٔ اول لبه‌های پائین و بالای پنجرهٔ ارسال یکی هستند، ولی با گذشت زمان موقعیت آنها طبق شکل تغییر می‌کند.

۳-٤-۱ پروتکل پنجرهٔ لغزندهٔ 1-بیتی

قبل از پرداختن به حالت کلی، اجازه دهیدابتدا پروتکلی با پنجرهٔ لغزندهٔ 1-بیتی را مورد بررسی قرار دهیم. در واقع این پروتکل نوعی پروتکل توقف-انتظار است، چون فرستنده قبل از گرفتن تصدیق دریافت فریم فرستاده شده، فریم بعدی را ارسال نخواهد کرد.

در شکل ۳-۱۴ پروتکل پنجرهٔ لغزندهٔ 1-بیتی را ملاحظه می‌کنید. مانند پروتکلهای قبلی، این پروتکل هم با تعریف متغیرها شروع می‌شود. متغیر *next_frame_to_send* فریم بعدیست که فرستنده باید بفرستد. در طرف مقابل هم، متغیر *frame_expected* فریمی را نشان می‌دهد که گیرنده منتظر آن است. در هر دو طرف، تنها حالتهای مجاز فقط 0 یا 1 است.

```
/* Protocol 4 (sliding window) is bidirectional. */
#define MAX_SEQ 1                          /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
```

```
{
  seq_nr next_frame_to_send;                    /* 0 or 1 only */
  seq_nr frame_expected;                        /* 0 or 1 only */
  frame r, s;                                   /* scratch variables */
  packet buffer;                                /* current packet being sent */
  event_type event;
  next_frame_to_send = 0;                       /* next frame on the outbound
stream */
  frame_expected = 0;                           /* frame expected next */
  from_network_layer(&buffer);                  /* fetch a packet from the network
layer */
  s.info = buffer;                              /* prepare to send the initial frame */
  s.seq = next_frame_to_send;                   /* insert sequence number into
frame */
  s.ack = 1 - frame_expected;                   /* piggybacked ack */
  to_physical_layer(&s);                        /* transmit the frame */
  start_timer(s.seq);                           /* start the timer running */
  while (true) {
      wait_for_event(&event);                   /* frame_arrival, cksum_err, or
timeout */
      if (event == frame_arrival) {             /* a frame has arrived
undamaged. */
            from_physical_layer(&r);            /* go get it */
            if (r.seq == frame_expected) {      /* handle inbound frame
stream. */
                  to_network_layer(&r.info);    /* pass packet to network
layer */
                  inc(frame_expected);          /* invert seq number expected
next */
  }
            if (r.ack == next_frame_to_send) {  /* handle outbound frame
stream. */
                  stop_timer(r.ack);            /* turn the timer off */
                  from_network_layer(&buffer);  /* fetch new pkt from network
layer */
                  inc(next_frame_to_send);      /* invert sender's sequence
number */
            }
      }
      s.info = buffer;                          /* construct outbound frame */
      s.seq = next_frame_to_send;               /* insert sequence number into it
```

```
*/
        s.ack = 1 - frame_expected;        /* seq number of last received
frame */
        to_physical_layer(&s);             /* transmit a frame */
        start_timer(s.seq);                /* start the timer running */
    }
  }
```

شکل ۳-۱۴. پروتکل پنجرهٔ لغزندهٔ 1-بیتی.

در شرایط عادی، یکی از دو طرف پیش دستی کرده و اولین فریم را می‌فرستد. بعبارت دیگر، فقط یکی از دو لایهٔ پیوند داده روالهای *to_physical_layer* و *start_timer* را خارج از حلقهٔ اصلی برنامه اجرا می‌کند. اگر در پیشامدی نادر هر دو طرف بطور همزمان شروع به ارسال اولین فریم کنند، وضعیت عجیبی پیش می‌آید، که بعداً آنرا توضیح خواهیم داد. ماشین شروع‌کننده اولین بسته را از لایهٔ شبکه گرفته، یک فریم از آن می‌سازد، و سپس ارسال می‌کند. وقتی این فریم (یا هر فریم دیگری) به طرف مقابل رسید، لایهٔ پیوند داده گیرنده چک می‌کند که تکراری نباشد (درست مثل پروتکل ۳). اگر این همان فریم موردنظر باشد، به لایهٔ شبکه تحویل شده و پنجرهٔ دریافت یک واحد به جلو لغزانده می‌شود.

فیلد تصدیق دریافت حاوی شمارهٔ آخرین فریمیست که بدون خطا دریافت شده است. اگر این شماره با شمارهٔ فریمی که فرستنده در صدد ارسال آن است یکی باشد، فرستنده می‌فهمد که دیگر نیازی به فریم داخل بافر ندارد و می‌تواند بستهٔ بعدی را از لایهٔ شبکه بگیرد. اگر شماره‌ها یکی نباشند، فرستنده باید به ارسال همان فریم قبلی ادامه دهد. وقتی یک فریم دریافت شود، فریمی نیز پس فرستاده می‌شود.

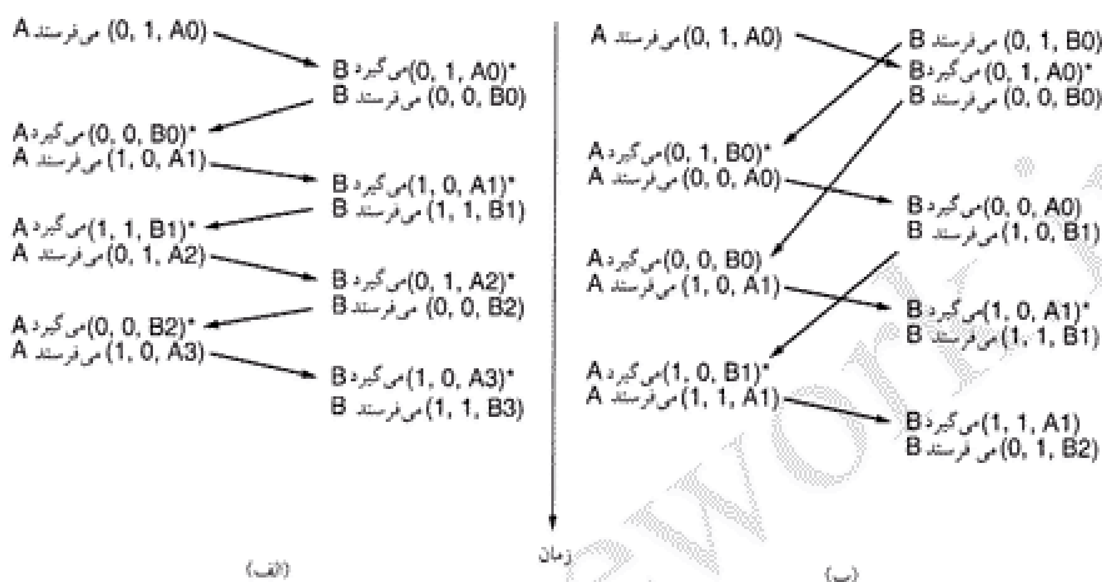
حال اجازه دهید ببینیم پروتکل ۴ در شرایط غیرعادی چگونه رفتار می‌کند. فرض کنید ماشین A درصدد ارسال فریم 0 به کامپیوتر B است، و در همان زمان B نیز تصمیم می‌گیرد فریم 0 خود را به A بفرستد. در ضمن فرض می‌کنیم که A مسابقه را زودتر شروع می‌کند، ولی فاصلهٔ زمانی تایمر آن بسیار کوتاه است. در نتیجه، ماشین A پشت سر هم فریمهای یکسان، با $seq = 0$ و $ack = 1$، به B می‌فرستد.

وقتی اولین فریم به کامپیوتر B رسید، پذیرفته شده و *frame_expected* به 1 ست می‌شود؛ تمام فریمهای بعدی رد خواهند شد، چون اینک B در انتظار فریمی با شمارهٔ ترتیبی 1 است نه 0. علاوه بر آن، چون تمام در فریمهای تکراری $ack = 1$ و B همچنان منتظر تصدیق دریافت فریم 0 است، ماشین B گرفتن بسته از لایهٔ شبکهٔ خود را متوقف خواهد کرد.

بعد از آن که تمام فریمهای تکراری به B رسیدند، B یک فریم با $seq = 0$ و $ack = 0$ به A می‌فرستد. بالاخره یکی از این فریمها سالم به A می‌رسد، و باعث می‌شود تا A ارسال فریم بعدی را شروع کند. همانطور که می‌بینید، در هیچ شرایطی بستهٔ تکراری به لایهٔ شبکه نمی‌رسد، بسته‌ای گم نمی‌شود، و سیستم قفل نمی‌کند.

اما اگر هر دو طرف در یک لحظه شروع به ارسال اولین فریم کنند، وضعیت عجیبی پیش می‌آید. این مشکل سنکرون شدن را در شکل ۳-۱۵ (ب) مشاهده می‌کنید. (در شکل ۳-۱۵ الف برای مقایسه عملکرد عادی پروتکل ۴ نشان داده شده است.) اگر B قبل از ارسال فریمهای خود منتظر دریافت اولین فریم A بماند، هیچ مشکلی پیش نمی‌آید و تمام بسته‌ها براحتی پذیرفته می‌شوند (شکل الف). ولی اگر هر دو با هم مخابرهٔ اولین فریم را شروع کنند، این فریمها با یکدیگر تصادم کرده، و حالت (ب) پیش می‌آید. در (الف) دریافت هر فریم باعث گرفتن یک بسته از لایهٔ شبکه می‌شود، و هیچ فریم تکراری هم وجود ندارد. اما در (ب) با وجود اینکه هیچ خطایی هم در

کانال وجود ندارد، نصف فریمها تکراری هستند. این وضعیت هنگامی که تایمر یکی از دو طرف بیش از حد کوتاه باشد، نیز پیش می‌آید (حتی اگر دو طرف همزمان شروع به مخابرهٔ فریمهای خود نکرده باشند). در چنین وضعیتی حتی امکان دارد برخی از فریمها سه یا چهار بار تکرار شوند.



شکل ۳-۱۵. دو سناریوی پروتکل ۴. (الف) حالت عادی. (ب) حالت غیرعادی. اعداد داخل پرانتز از چپ به راست عبارتند از: seq ، ack و شمارهٔ بسته. بسته‌هایی که پذیرفته و به لایهٔ شبکه تحویل می‌شوند، با * مشخص شده‌اند.

۳-٤-۲ پروتکل «N تا به عقب برگرد»

تا اینجا بطور ضمنی فرض کرده بودیم که زمان رسیدن فریم به مقصد بعلاوهٔ زمان برگشت فریم تصدیق ناچیز است. اما گاهی این فرض آشکارا نادرست است. در چنین مواردی زمان طولانی رفت و برگشت فریم می‌تواند تأثیر چشمگیری روی کارایی مصرف پهنای باند داشته باشد. بعنوان مثال، یک کانال ماهواره‌ای با پهنای باند 50 kbps و زمان تأخیر رفت و برگشت 500 msec را در نظر بگیرید. فرض کنید می‌خواهیم با این لینک ماهواره‌ای و با استفاده از پروتکل ۴ فریمهای 1000 بیتی ارسال کنیم. در لحظهٔ $t = 0$ فرستنده مخابرهٔ اولین فریم را شروع می‌کند، و در $t = 20$ msec کار ارسال فریم به پایان می‌رسد. اما در بهترین شرایط (و با فرض اینکه در گیرنده نیز هیچ تأخیری وجود ندارد)، تا $t = 270$ msec این فریم هنوز به گیرنده نرسیده، و تا $t = 520$ msec نیز مسلماً فریم تصدیق دریافت به دست فرستنده نخواهد رسید. این بدان معناست که فرستنده در 500/520 یا 96% زمان باید متوقف بماند، و نمی‌تواند چیزی بفرستد؛ بعبارت دیگر، از فقط 4% پهنای باند استفاده می‌کند. پیداست که ترکیب تأخیر طولانی در کانال، پهنای باند زیاد، و فریمهای کوچک چیزی جز اتلاف وحشتناک منابع نیست.

مشکلی که در بالا دیدید از آنجا ناشی می‌شد که فرستنده برای ارسال یک فریم باید منتظر تصدیق دریافت فریم قبلی بماند. اما اگر این قید را برداریم، می‌توانیم به کارایی بهتری دست پیدا کنیم. در واقع بجای 1 فریم، معمولاً فرستنده می‌تواند w فریم بفرستد و پس از آن منتظر رسیدن فریمهای تصدیق بماند. اگر w طوری انتخاب شود که فرستنده در تمام مدت زمان تأخیر رفت و برگشت در حال ارسال فریم باشد، می‌تواند بدون مشکل از تمام پهنای باند استفاده کند. در مثال بالا w باید حداقل 26 باشد. فرستنده مانند قبل ارسال اولین فریم را در $t = 0$ شروع می‌کند، و زمانی که فریم 26 را می‌فرستد ($t = 520$ msec)، تصدیق فریم 0 را دریافت خواهد کرد. از آن به

بعد نیز فریمهای تصدیق دریافت هر 20 msec از راه می‌رسند، و فرستنده می‌تواند بطور پیوسته به ارسال فریمها ادامه دهد. در تمام زمانها فرستنده 25 یا 26 فریم در بافر خود دارد که هنوز تصدیق دریافت آنها را نگرفته است. به بیان دیگر، اندازهٔ پنجره ارسال حداکثر 26 است.

پنجرهٔ ارسال بزرگ فقط زمانی لازم می‌شود که حاصلضرب **پهنای باند** × **تأخیر رفت و برگشت** عددی بزرگ باشد. اگر پهنای باند زیاد باشد، حتی با تأخیر کم نیز فرستنده بسرعت پنجرهٔ ارسال را پُر می‌کند. اگر تأخیر رفت و برگشت زیاد باشد (مانند کانالهای ماهواره‌ای GEO)، حتی در پهنای باند متوسط نیز پنجرهٔ ارسال بزودی پُر می‌شود. ظرفیت یک کانال اساساً با حاصلضرب این دو عامل (پهنای باند و تأخیر رفت و برگشت) تعیین می‌شود، و فرستنده برای رسیدن به حداکثر کارایی باید بتواند کانال را بدون وقفه پُر کند.

به این تکنیک **لوله‌کشی** (pipelining) گفته می‌شود. اگر ظرفیت کانال b bits/sec، اندازهٔ فریم l بیت، و تأخیر رفت و برگشت R sec باشد، زمان لازم برای ارسال هر فریم l/b sec خواهد بود. بعد از ارسال آخرین بیت یک فریم، و قبل از رسیدن این بیت به گیرنده، تأخیری بمیزان $R/2$ وجود دارد؛ بازگشت فریم تصدیق دریافت نیز با همین مقدار تأخیر همراه است (که کل تأخیر به R می‌رسد). با پروتکل توقف-انتظار، خط بمدت l/b کار کرده و سپس بمدت R بیکار می‌ماند، که در نتیجه

$$\text{بهرهٔ خط} = l / (l + bR)$$

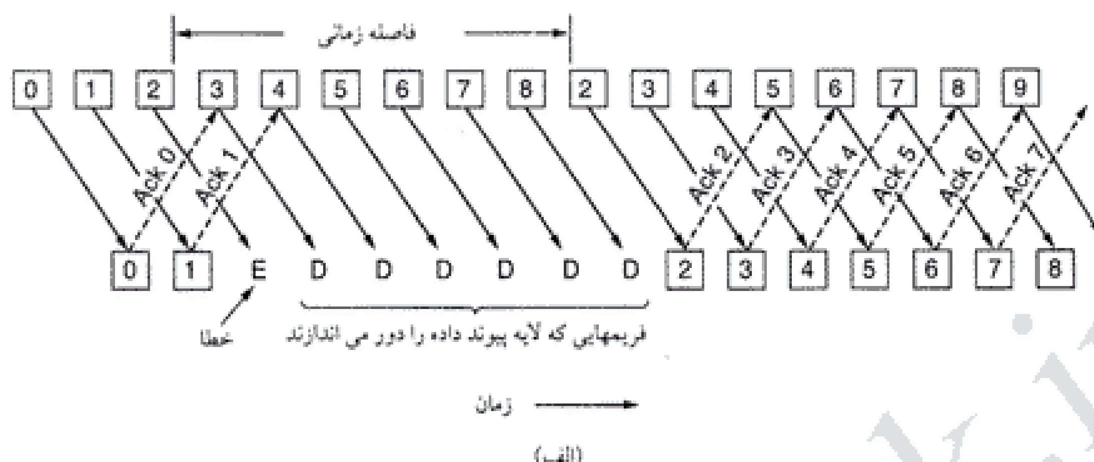
اگر $l < bR$، کارایی خط زیر 50% خواهد بود. از آنجائیکه تأخیر رفت و برگشت خطوط انتقال هرگز صفر نیست، با تکنیک لوله‌کشی می‌توان کاری کرد که خط همیشه مشغول باشد - اما اگر مقدر این تأخیر ناچیز باشد، ارزش پیچیدگی بیشتر پروتکل را ندارد.

لوله‌کشی فریمها در کانالهای غیرقابل‌اطمینان می‌تواند منجر به مشکلات جدی شود. اول اینکه، اگر یکی از فریمهای این صف طویل ناپدید یا خراب شود، چه خواهد شد؟ قبل از آنکه حتی فرستنده متوجه این خطا شود، تعداد زیادی از فریمهای بعدی به گیرنده رسیده‌اند. با رسیدن فریم خراب، گیرنده مسلماً آنرا دور می‌اندازد، اما با فریمهای سالم بعدی چه باید بکند؟ بیاد داشته باشید که لایهٔ پیوند داده باید بسته‌ها را با ترتیب صحیح به لایهٔ شبکه تحویل دهد. در شکل ۳-۱۶ این وضعیت (بروز خطا در **خط لوله** - pipeline) را ملاحظه می‌کنید. اجازه دهید آنرا دقیقتر بررسی کنیم.

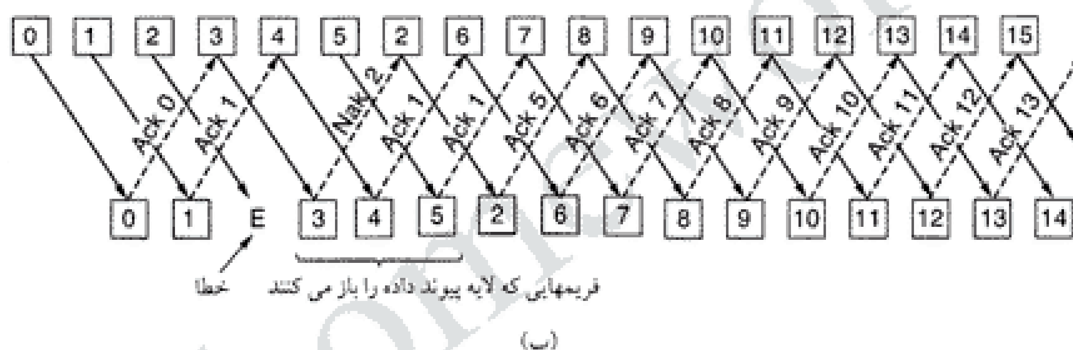
برای مقابله با خطا در تکنیک لوله‌کشی دو رهیافت کلی وجود دارد. در رهیافت اول، که به «N تا به عقب برگرد» معروف است، گیرنده تمام فریمهای بعد از فریم خراب را دور می‌اندازد و هیچ تصدیقی برای آنها برنمی‌گرداند. این استراتژی معادل است با پنجرهٔ دریافتی باندازهٔ 1. بعبارت دیگر، لایهٔ پیوند داده در گیرنده هیچ فریمی غیر ازآن فریمی که باید به لایهٔ شبکه تحویل دهد، را قبول نمی‌کند. اگر پنجرهٔ ارسال فرستنده قبل از انقضای تایمر پُر شود، خط لوله شروع به خالی شدن خواهد کرد. پس از آن تایمر فرستنده به انتها رسیده و تمام فریمهای باقی مانده (از فریمی که خراب یا گم شده) را دوباره ارسال خواهد کرد. اگر نرخ خطا در خط زیاد باشد (مانند کانالهای بیسیم)، این رهیافت باعث اتلاف شدید پهنای باند خواهد شد.

در شکل ۳-۱۶ (الف) این حالت را ملاحظه می‌کنید. در این مثال فریمهای 0 و 1 سالم به مقصد رسیده‌اند، ولی فریم 2 خراب شده است. فرستنده تا زمانی که تایمر این فریم منقضی نشود، از خراب شدن آن مطلع نخواهد شد. پس از آنکه فرستنده فهمید فریم 2 سالم به مقصد نرسیده، برمی‌گردد و ارسال فریمها را از این فریم از سر می‌گیرد.

رهیافت دوم مقابله با خطا در تکنیک لوله‌کشی **تکرار انتخابی** (selective repeat) نام دارد. در این رهیافت، فریم خراب در گیرنده دور انداخته شده، ولی فریمهای سالم بعدی بافر می‌شوند. وقتی تایمر فریم معیوب منقضی



(الف)



(ب)

شکل ۳-۱۶. مقابله با خطا در خط لوله. تأثیر خطا وقتی که (الف) اندازهٔ پنجرهٔ دریافت گیرنده 1 است، و (ب) پنجرهٔ دریافت بزرگ است.

شد، فرستنده فقط همین فریم (که قدیمی‌ترین فریم تصدیق نشده است) را مجدداً ارسال می‌کند. اگر این فریم سالم به مقصد رسید، گیرنده این فریم و فریمهای بافر شده را بترتیب به لایهٔ شبکه تحویل می‌دهد. در رهیافت تکرار انتخابی، معمولاً گیرنده برای فریمهای خراب (فریمهایی که جمع تطبیقی آنها اشتباه است، یا خارج از نظم وارد می‌شوند) نیز یک فریم **تصدیق دریافت منفی** (negative acknowledgement) – که به فریم NAK معروف است – به فرستنده برمی‌گرداند. فریم NAK کارایی سیستم را به مقدار زیادی بهبود می‌بخشد، چون باعث می‌شود که فرستنده قبل از انقضای تایمر کار ارسال مجدد فریمهای از دست رفته را شروع کند.

در شکل ۳-۱۶ (ب) نیز فریمهای 0 و 1 سالم به مقصد رسیده‌اند، ولی فریم 2 خراب شده است. وقتی فریم 3 به گیرنده می‌رسد، لایهٔ پیوند داده متوجه می‌شود که یک فریم جا افتاده است، پس یک NAK برای فریم 2 به فرستنده فرستاده، و فریم 3 را در بافری که برای این منظور اختصاص یافته، ذخیره می‌کند. فریمهای 4 و 5 نیز پس از رسیدن به گیرنده، بجای تحویل به لایهٔ شبکه، بافر می‌شوند. با رسیدن NAK فریم 2 ، فرستنده بلافاصله این فریم را ارسال می‌کند، که با رسیدن آن به مقصد، گیرنده می‌تواند فریم 2 و فریمهای پس از آن (تا فریم 5) را به لایهٔ شبکه تحویل دهد (و تصدیق دریافت آنها را به فرستنده برگرداند). حتی اگر NAK فریم 2 در راه گم شود و به دست فرستنده نرسد، باز هم پس از انقضای تایمر، فرستنده نسبت به ارسال مجدد آن (و فقط همین یک فریم) اقدام خواهد کرد؛ که البته در این میان فقط وقت بیشتری تلف خواهد شد. در واقع، NAK فقط ارسال مجدد

فریمهای معیوب را تسریع می‌کند.

رهیافت تکرار انتخابی خاص پنجره‌های دریافت بزرگتر از 1 است. هر فریمی در داخل پنجرهٔ دریافت قرار داشته باشد، می‌تواند بافر شود تا فریمهای قبل از آن دریافت و به لایهٔ شبکه تحویل شوند. البته اگر پنجرهٔ دریافت خیلی بزرگ باشد، لایهٔ پیوند داده به حافظهٔ زیادی برای بافر کردن فریمها نیاز دارد.

این دو رهیافت داد و ستدی هستند بین پهنای باند و فضای بافر لایهٔ پیوند داده، که بسته به اهمیت هر یک از این منابع می‌توان یکی از رهیافت‌های «N تا به عقب برگرد» یا «تکرار انتخابی» را انتخاب کرد. در شکل ۳-۱۷ پروتکل لوله‌کشی با پنجرهٔ دریافت 1 را ملاحظه می‌کنید؛ در این پروتکل تمام فریمهای بعد از یک فریم معیوب دور انداخته می‌شوند. در این پروتکل برای اولین بار فرض نامحدود بودن بسته‌هایی که لایهٔ شبکه می‌تواند ارائه کند، را نیز کنار گذاشته‌ایم. در اینجا، وقتی لایهٔ شبکه آمادهٔ ارسال یک بسته است، رویداد *network_layer_ready* را تحریک می‌کند. با این حال، برای آنکه هیچوقت بیش از *MAX_SEQ* فریم وارد صف ارسال لایهٔ پیوند داده نشود، باید کاری کنیم که این لایه بتواند بطریقی مانع ارسال بسته‌های اضافی از لایهٔ شبکه شود - برای این منظور از روالهای کتابخانه‌ای *enable_network_layer* و *disable_network_layer* استفاده کرده‌ایم.

```
/* Protocol 5 (go back n) allows multiple outstanding frames. The sender may transmit up
   to MAX_SEQ frames without waiting for an ack. In addition, unlike in the previous
   protocols, the network layer is not assumed to have a new packet all the time. Instead,
the network layer causes a network_layer_ready event when there is a packet to send.
*/

#define MAX_SEQ 7                              /* should be 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Return true if a <= b < c circularly; false otherwise. */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
        return(true);
    else
        return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])

{
/* Construct and send a data frame. */
  frame s;                                  /* scratch variable */

  s.info = buffer[frame_nr];                   /* insert packet into frame */
```

```
  s.seq = frame_nr;                    /* insert sequence number into frame */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);        /* piggyback ack */
  to_physical_layer(&s);               /* transmit the frame */
  start_timer(frame_nr);                    /* start the timer running */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;                /* MAX_SEQ > 1; used for outbound
stream */
  seq_nr ack_expected;                 /* oldest frame as yet unacknowledged */
  seq_nr frame_expected;                    /* next frame expected on inbound
stream */
  frame r;                             /* scratch variable */
  packet buffer[MAX_SEQ + 1];               /* buffers for the outbound stream */
  seq_nr nbuffered;                         /* # output buffers currently in use */
  seq_nr i;                                 /* used to index into the buffer array */
  event_type event;

  enable_network_layer();                   /* allow network_layer_ready events */
  ack_expected = 0;                         /* next ack expected inbound */
  next_frame_to_send = 0;                   /* next frame going out */
  frame_expected = 0;                       /* number of frame expected inbound */
  nbuffered = 0;                       /* initially no packets are buffered */
  while (true) {
     wait_for_event(&event);                /* four possibilities: see event_type
above */

     switch(event) {
       case network_layer_ready:            /* the network layer has a packet to send
*/
            /* Accept, save, and transmit a new frame. */
             from_network_layer(&buffer[next_frame_to_send]); /* fetch new packet */
            nbuffered = nbuffered + 1;           /* expand the sender's window */
             send_data(next_frame_to_send, frame_expected, buffer); /* transmit the
frame */
             inc(next_frame_to_send);            /* advance sender's upper window
edge */
            break;

       case frame_arrival:                  /* a data or control frame has arrived */
```

```
            from_physical_layer(&r);          /* get incoming frame from physical
layer */

            if (r.seq == frame_expected) {
                  /* Frames are accepted only in order. */
                  to_network_layer(&r.info);     /* pass packet to network layer */
                  inc(frame_expected);           /* advance lower edge of receiver's
window */
             }

             /* Ack n implies n - 1, n - 2, etc.  Check for this. */
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                  /* Handle piggybacked ack. */
                  nbuffered = nbuffered - 1; /* one frame fewer buffered */
                  stop_timer(ack_expected); /* frame arrived intact; stop timer */

                  inc(ack_expected);          /* contract sender's window */
            }
            break;

      case cksum_err: break;                  /* just ignore bad frames */

      case timeout:                           /* trouble; retransmit all outstanding
frames */
            next_frame_to_send = ack_expected;    /* start retransmitting here */
            for (i = 1; i <= nbuffered; i++) {
                  send_data(next_frame_to_send, frame_expected, buffer); /* resend
1 frame */
                  inc(next_frame_to_send);      /* prepare to send the next one */
            }

    }

    if (nbuffered < MAX_SEQ)
;)(reyal_krowten_elbane
     else
            disable_network_layer();
  }
}
```

شکل ۳-۱۷. پروتکل پنجرهٔ لغزندهٔ n تا به عقب برگرده.

توجه کنید که در هر لحظه نباید بیش از MAX_SEQ فریم (و نه MAX_SEQ + 1) در صف ارسال وجود داشته باشد - حتی با وجود اینکه تعداد شماره‌های ترتیبی، در MAX_SEQ + 1 است: 0, 1, 2, ..., MAX_SEQ. برای پی بردن به علت این محدودیت، سناریوی زیر را که در آن MAX_SEQ = 7 نظر بگیرید:

۱. فرستنده فریمهای 0 تا 7 (هشت فریم) را ارسال می‌کند.
۲. پس از مدتی، تصدیق دریافت فریم 7 (با استفاده از تکنیک سواری‌مجانی) به فرستنده برمی‌گردد.
۳. فرستنده ۸ فریم بعدی را با همان شماره‌های ترتیبی 0 تا 7 ارسال می‌کند.
٤. حال فریم تصدیق دریافت دیگری برای فریم شمارهٔ 7 (به همان صورت سواری مجانی) از راه می‌رسد.

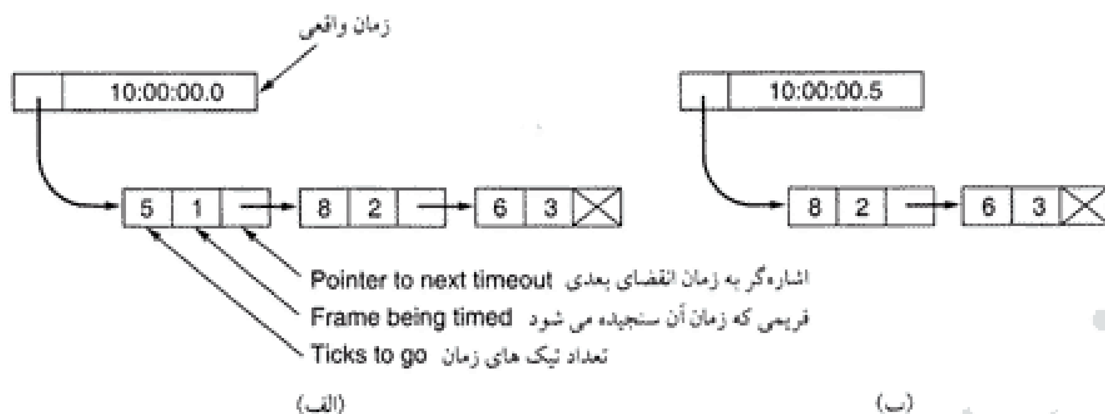
سؤال این است: آیا تمام ۸ فریمی که متعلق به دسته دوم بودند، سالم به مقصد رسیده‌اند، یا (با توجه به اینکه تمام فریمهای پس از یک فریم خراب دور انداخته می‌شوند) همگی از بین رفته‌اند؟ اگر دقت کرده باشید، در هر دو حالت گیرنده تصدیق دریافت فریم 7 را پس می‌فرستد، و فرستنده هم راهی برای تشخیص این موضوع ندارد. به همین دلیل حداکثر فریمهایی که در صف ارسال می‌ایستند، باید MAX_SEQ باشد.

با اینکه پروتکل ۵ فریمهای پس از یک فریم معیوب را بافر نمی‌کند، اما هنوز به مقداری بافر در سمت فرستنده نیاز داریم. از آنجائیکه فرستنده باید تمام فریمهای داخل پنجرهٔ ارسال را تا رسیدن تصدیق دریافت آنها نگه دارد (چون ممکنست لازم شود آنها را دوباره بفرستد)، باید فضای کافی برای بافر کردن این فریمها داشته باشد. در این پروتکل وقتی تصدیق دریافت فریم n از راه برسد، فریمهای $n - 1$، $n - 2$ و ... نیز بطور خودکار تصدیق شده محسوب می‌شوند. این ویژگی بویژه اگر فریمهای تصدیق قبلی در راه برگشت به فرستنده از بین بروند، اهمیت می‌یابد. با رسیدن هر فریم تصدیق دریافت، لایهٔ پیوند داده چک می‌کند که کدام بافرها را می‌تواند آزاد کند. وقتی بافر آزاد (و جایی در پنجرهٔ ارسال باز) شد، لایهٔ شبکه (که قبلاً متوقف شده) دوباره با رویداد *enable_network_layer* فعال می‌شود و می‌تواند بسته‌های بعدی را به لایهٔ پیوند داده بفرستد.

در پروتکل ۵ فرض کرده‌ایم که همیشه ترافیک برگشتی کافی برای سواری‌مجانی وجود دارد. اگر چنین نباشد، فریمهای تصدیق دریافت را هم نمی‌توان ارسال کرد. در پروتکل ۴ چنین فرضی وجود نداشت، و برای هر فریم یک فریم تصدیق دریافت مستقل پس فرستاده می‌شد. در پروتکل آینده این مشکل را هم بنحو جالبی حل خواهیم کرد.

دیدید که در پروتکل ۵، فرستنده می‌تواند در هر لحظه تعداد زیادی فریم ارسال شده (ولی هنوز تصدیق نشده) در بافر خود داشته باشد: هر یک از این فریمها یک تایمر مستقل می‌خواهند. این تایمرها را می‌توان بصورت نرم‌افزاری و با استفاده از وقفه‌های ساعت سخت‌افزاری، ایجاد کرد. این تایمرها تشکیل یک لیست **پیوندی** (linked list) می‌دهند، که هر گره آن سه بخش دارد: زمان باقی ماندهٔ تایمر، فریمی که به این تایمر مربوط است، و یک اشاره‌گر به گره بعدی.

در شکل ۳-۱۸ (الف) طرز پیاده‌سازی این تایمرها را می‌بینید. فرض کنید که ساعت سیستم هر 100 msec یک **تیک** (وقفهٔ سخت‌افزاری) می‌فرستد. در لحظهٔ شروع، که زمان واقعی 10:00:00.0 است، سه تایمر برای زمانهای 10:00:00.5 ، 10:00:01.3 و 10:00:01.9 ست می‌شوند. با هر تیک ساعت سخت‌افزاری، شمارندهٔ تایمری که در رأس لیست قرار دارد، کاهش می‌یابد. وقتی این شمارنده 0 شد، گره مربوطه از لیست حذف می‌شود؛ شکل ۳-۱۸(ب) را ببینید. سازماندهی تایمرها به صورت فوق باعث می‌شود که در هر بار فراخوانی روالهای *start_timer* و *stop_timer* کل لیست اسکن شود، ولی از سوی دیگر کار لازم برای به روز در آوردن تایمرها در هر تیک بسیار ناچیز است. همانطور که می‌بینید، در پروتکل ۵ روالهای *start_timer* و *stop_timer* پارامتری می‌گیرند، که نشان می‌دهد زمان کدام فریم بایستی سنجیده شود.



شکل ۳-۱۸. شبیه‌سازی چند تایمر بوسیلهٔ نرم‌افزار.

۳-٤-۳ پروتکل تکرار انتخابی

اگر نرخ خطا ناچیز باشد، پروتکل ۵ بخوبی کار خواهد کرد، اما در خطوط پُرنویز این پروتکل مقدار زیادی از پهنای باند را (برای ارسال مجدد فریمها) به هدر می‌دهد. استراتژی دیگر مقابله با خطاهای خط اینست که به گیرنده اجازه دهیم فریمهای سالمی که بعد از یک (یا چند) فریم معیوب از راه می‌رسند، را بافر کند. چنین پروتکلی دیگر فریمها را به صرف اینکه فریم قبلی آنها خراب بوده یا گم شده، دور نمی‌اندازد.

در این پروتکل، فرستنده و گیرنده هر دو پنجره‌ای از شماره‌های قابل قبول دارند. پنجرهٔ ارسال در فرستنده از 0 شروع شده، و می‌تواند تا *MAX_SEQ* بالا برود؛ اما اندازهٔ پنجرهٔ دریافت همیشه ثابت، و معادل *MAX_SEQ* است. گیرنده برای هر یک از فریمهایی که در پنجرهٔ دریافت قرار می‌گیرند، یک بافر ثابت دارد، که هر یک از این بافرها دارای بیتی هستند (بنام بیت *arrived*) که مشخص می‌کند بافر پُر است یا خالی. وقتی یک فریم بدست گیرنده می‌رسد، شمارهٔ آن را با تابع *between* چک می‌کند تا ببیند در داخل پنجرهٔ دریافت است یا خارج آن. اگر فریم داخل پنجره باشد و تکراری هم نباشد، پذیرفته شده و در بافر ذخیره می‌شود (خواه این فریم محتوی بسته‌ای باشد که باید در نوبت بعد به لایهٔ شبکه داده شود، یا نه). البته این فریم تا زمانی که تمام فریمهای قبل از آن به لایهٔ شبکه تحویل نشده‌اند، در لایهٔ پیوند داده می‌ماند. در شکل ۳-۱۹ پروتکلی که بر اساس این الگوریتم نوشته شده، را ملاحظه می‌کنید.

```
    /* Protocol 6 (selective repeat) accepts frames out of order but passes packets to the
    network layer in order. Associated with each outstanding frame is a timer. When the
timer
    expires, only that frame is retransmitted, not all the outstanding frames, as in
protocol 5. */

    #define MAX_SEQ 7                          /* should be 2^n - 1 */
    #define NR_BUFS ((MAX_SEQ + 1)/2)
    typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
event_type;
    #include "protocol.h"
    boolean no_nak = true;                     /* no nak has been sent yet */
```

```
    seq_nr oldest_frame = MAX_SEQ + 1;          /* initial value is only for the simulator
*/
    static boolean between(seq_nr a, seq_nr b, seq_nr c)
    {
    /* Same as between in protocol5, but shorter and more obscure. */
      return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
    }
    static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected,
packet buffer[])
    {
    /* Construct and send a data, ack, or nak frame. */
      frame s;                                 /* scratch variable */

      s.kind = fk;                             /* kind == data, ack, or nak */
      if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
      s.seq = frame_nr;                        /* only meaningful for data frames */
      s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
      if (fk == nak) no_nak = false;               /* one nak per frame, please */
      to_physical_layer(&s);                   /* transmit the frame */
      if (fk == data) start_timer(frame_nr % NR_BUFS);
      stop_ack_timer();                            /* no need for separate ack frame */
    }
    void protocol6(void)
    {
      seq_nr ack_expected;                     /* lower edge of sender's window */
      seq_nr next_frame_to_send;                   /* upper edge of sender's window +
1 */
      seq_nr frame_expected;                       /* lower edge of receiver's window */
      seq_nr too_far;                          /* upper edge of receiver's window + 1
*/
      int i;                                   /* index into buffer pool */
      frame r;                                 /* scratch variable */
      packet out_buf[NR_BUFS];                     /* buffers for the outbound stream */
      packet in_buf[NR_BUFS];                  /* buffers for the inbound stream */
      boolean arrived[NR_BUFS];                    /* inbound bit map */
      seq_nr nbuffered;                            /* how many output buffers currently
used */
      event_type event;

      enable_network_layer();                      /* initialize */
      ack_expected = 0;                            /* next ack expected on the inbound
```

```
stream */
        next_frame_to_send = 0;                 /* number of next outgoing frame */
        frame_expected = 0;
        too_far = NR_BUFS;
        nbuffered = 0;                       /* initially no packets are buffered */
        for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
        while (true) {
            wait_for_event(&event);             /* five possibilities: see event_type
above */
            switch(event) {
              case network_layer_ready:         /* accept, save, and transmit a new
frame */
                    nbuffered = nbuffered + 1;       /* expand the window */
      from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]);    /* fetch
new packet */
     send_frame(data, next_frame_to_send, frame_expected, out_buf);    /*
transmit the frame */
                    inc(next_frame_to_send);         /* advance upper window edge */
                    break;

              case frame_arrival:               /* a data or control frame has arrived
*/
                    from_physical_layer(&r);         /* fetch incoming frame from
physical layer */
                    if (r.kind == data) {
                         /* An undamaged frame has arrived. */
                         if ((r.seq != frame_expected) && no_nak)
     send_frame(nak, 0, frame_expected, out_buf);    else
start_ack_timer();
    if (between(frame_expected, r.seq, too_far) &&
(arrived[r.seq%NR_BUFS] == false)) {
                              /* Frames may be accepted in any order. */
                              arrived[r.seq % NR_BUFS] = true;   /* mark buffer as
full */
                              in_buf[r.seq % NR_BUFS] = r.info;  /* insert data into
buffer */
                              while (arrived[frame_expected % NR_BUFS]) {
                                   /* Pass frames and advance window. */
                               to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                                   no_nak = true;
                                   arrived[frame_expected % NR_BUFS] = false;
```

```
        inc(frame_expected);    /* advance lower edge of receiver's window */
                        inc(too_far);    /* advance upper edge of receiver's window */
                        start_ack_timer();   /* to see if a separate ack is needed */
                    }
                }
            }
        if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
        send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

            while (between(ack_expected, r.ack, next_frame_to_send)) {
                nbuffered = nbuffered - 1;    /* handle piggybacked ack */
                stop_timer(ack_expected % NR_BUFS); /* frame arrived intact */
                inc(ack_expected);      /* advance lower edge of sender's window */
            }
            break;

        case cksum_err:
            if (no_nak) send_frame(nak, 0, frame_expected, out_buf);/*  damaged frame */
            break;

        case timeout:
            send_frame(data, oldest_frame, frame_expected, out_buf); /*  we timed out */
            break;

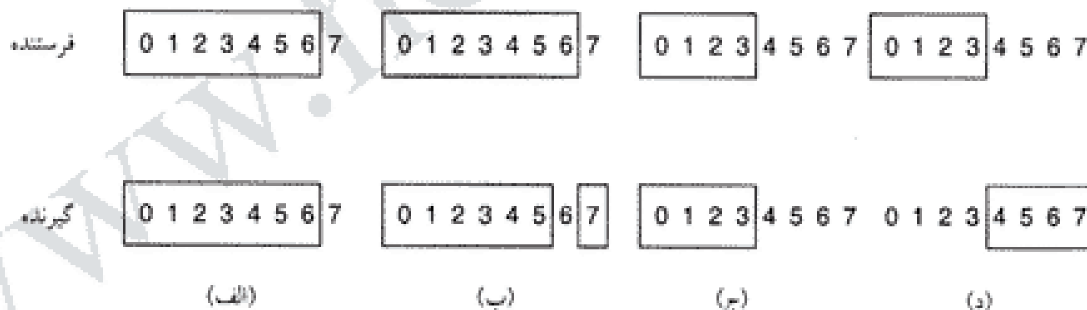
        case ack_timeout:
            send_frame(ack,0,frame_expected, out_buf);     /* ack timer expired; send ack */
        }
    ;)(reyal_krowten_elbasid esle ;)(reyal_krowten_elbane )SFUB_RN < dereffubn( fi
    }
}
```

شکل ۳-۱۹. پروتکل پنجرهٔ لغزندهٔ «تکرار انتخابی».

دریافت نامنظم فریمها مسائلی را بوجود می‌آورد که در پروتکلهای قبلی (که فریمها را فقط بترتیب شماره قبول می‌کنند) وجود نداشت. با یک مثال بهتر می‌توان این مشکل را نشان داد. فرض کنید از شماره‌های ترتیبی سه‌بیتی استفاده می‌کنیم، بنابراین فرستنده قبل از توقف برای رسیدن اولین فریم تصدیق دریافت می‌تواند حداکثر هفت فریم ارسال کند. در لحظهٔ شروع، پنجره‌های ارسال و دریافت مانند شکل ۳-۲۰ (الف) هستند. فرستنده فریمهای 0 تا 6 را می‌فرستد. پنجرهٔ دریافت گیرنده فقط اجازهٔ پذیرش فریمهایی را می‌دهد که شمارهٔ آنها (منحصراً) بین 0 تا 6 باشد. تمام هفت فریم اول سالم به مقصد می‌رسند، بنابراین گیرنده دریافت آنها را تصدیق کرده، پنجرهٔ دریافت را برای دریافت سری بعدی فریمها (7 ، 0 ، 1 ، 2 ، 3 ، 4 ، یا 5) بجلو می‌برد، و تمام بافرها را هم با علامت «خالی» نشانه‌گذاری می‌کند؛ شکل ۳-۲۰ (ب) را ببینید.

درست در همین لحظه یک صاعقه به خط تلفن اصابت کرده، و تمام فریمهای تصدیق دریافت را (در راه بازگشت به فرستنده) از بین می‌برد. پس از مدتی تایمر فریم 0 به انتها رسیده، و فرستنده این فریم را مجدداً ارسال می‌کند. وقتی این فریم تکراری به گیرنده رسید، گیرنده چک می‌کند که آیا در داخل پنجرهٔ دریافت هست یا خیر. متأسفانه همانطور که در شکل ۳-۲۰ (ب) می‌بینید، فریم 0 هنوز در داخل پنجرهٔ دریافت قرار دارد، بنابراین گیرنده آنرا قبول می‌کند. از آنجائیکه فریمهای 0 تا 6 دریافت شده‌اند، گیرنده تصدیق دریافت فریم 6 را سوار فریم بعدی کرده و به فرستنده برمی‌گرداند.

فرستنده هم خوشحال از اینکه تمام فریمهای فرستاده شده سالم به مقصد رسیده‌اند، پنجرهٔ ارسال را بجلو برده و بلافاصله فریمهای 7 ، 0 ، 1 ، 2 ، 3 ، 4 و 5 را می‌فرستد. وقتی فریم 7 به مقصد رسید، لایهٔ پیوند داده گیرنده آنرا تحویل لایهٔ شبکه می‌دهد. بلافاصله پس از آن لایهٔ پیوند داده چک می‌کند که آیا بستهٔ 0 معتبری وجود دارد یا خیر، و چون چنین بسته‌ای در بافرهای آن موجود است، آنرا به لایهٔ شبکه می‌دهد؛ در حالیکه می‌دانیم این همان بستهٔ 0اول است و نباید دوباره به لایهٔ شبکه داده شود - پس پروتکل ما مرتکب خطا شده است.



شکل ۳-۲۰. (الف) پنجره‌های ارسال و دریافت هفت‌تایی در لحظهٔ شروع. (ب) بعد از رسیدن هفت فریم به مقصد، و قبل از بازگشت تصدیق دریافت به فرستنده. (ج) پنجره‌های ارسال و دریافت چهارتایی در لحظهٔ شروع. (د) بعد از رسیدن چهار فریم به مقصد، و قبل از بازگشت تصدیق دریافت به فرستنده.

منشأ این مشکل آنجاست که بعد از جلو رفتن پنجرهٔ دریافت در گیرنده، شماره‌های معتبر جدید با شماره‌های قدیمی **همپوشانی** (overlap) دارد. در نتیجه، فریمهای بعدی می‌توانند تکراری باشند (اگر تمام فریمهای تصدیق دریافت از بین برود) یا نباشند (اگر تمام فریمهای تصدیق دریافت سالم به فرستنده برسند). گیرندهٔ بیچاره هیچ راهی برای تشخیص این وضعیت ندارد.

راه چارهٔ این معضل آن است که مطمئن شویم بعد از جلو رفتن پنجرهٔ گیرنده، با پنجرهٔ اصلی همپوشانی نداشته باشد. برای رسیدن به این هدف پنجرهٔ دریافت باید از نصف تعداد شماره‌های ترتیبی تجاوز نکند؛ شکل

۳-۲۰ (پ) و (ت) را ببینید. برای مثال، اگر از شماره‌های ترتیبی ۴-بیتی استفاده کنیم، محدودهٔ ما 0 تا 15 خواهد بود، و فرستنده نباید در هر لحظه بیش از هشت فریم تصدیق نشده در بافر خود داشته باشد. بدین‌ترتیب، وقتی گیرنده فریمهای 0 تا 7 را گرفته و پنجرهٔ دریافت را جلو ببرد، به سری 8 تا 15 می‌رسد که آشکار است دیگر با مشکل قبلی مواجه نخواهد شد. در حالت کلی، اندازهٔ پنجره در پروتکل ۶ بایستی حداکثر $(MAX_SEQ + 1)/2$ باشد. در مثال قبل، پنجرهٔ ارسال و دریافت را باید $4 = (7 + 1)/2$ انتخاب کنیم.

و حالا یک سؤال جالب دیگر: گیرنده چند بافر باید داشته باشد؟ گیرنده در هیچ شرایطی فریمی که شمارهٔ آن کمتر از لبهٔ پائین پنجرهٔ دریافت یا بیشتر از لبهٔ بالای آن باشد، را قبول نخواهد کرد. در نتیجه، تعداد بافرهای گیرنده باید معادل اندازهٔ پنجرهٔ دریافت باشد (نه تعداد شماره‌های ترتیبی). با شماره‌های ترتیبی ۴-بیتی، گیرنده به حداکثر هشت بافر نیاز دارد. وقتی فریم i بدست گیرنده می‌رسد، آنرا در بافری بشمارهٔ i MOD 8 قرار می‌دهد. شاید حدس زده باشید که فریمهای i و $i + 8$ هر دو در یک بافر قرار می‌گیرند، ولی توجه کنید که این دو فریم هرگز در یک پنجره واقع نمی‌شوند (چون برای آنکه چنین اتفاقی بیفتد، اندازهٔ پنجره باید حداقل 9 باشد).

به دلیل مشابه، تعداد تایمرهای فرستنده نیز باید معادل پنجرهٔ دریافت باشد، نه تعداد شماره‌های ترتیبی (چون هر تایمر به یک بافر وابسته است و وقتی تایمر به انتها می‌رسد، بافر دوباره ارسال می‌شود).

در پروتکل ۵ این پیش‌فرض ضمنی را داشتیم که بار کانال بسیار زیاد است: وقتی یک فریم از راه می‌رسد، تصدیق آن بلافاصله برگردانده نمی‌شود تا سوار فریم بعدی (که از گیرنده به فرستنده می‌رود) شود. اما اگر ترافیک در جهت مخالف کم باشد، فریمهای تصدیق دریافت خیلی معطل خواهند شد. در این روش، وقتی فرستنده *MAX_SEQ* بسته فرستاد، منتظر می‌ماند و از آنجائیکه ترافیک جهت مقابل کم است، مدت زیادی بیکار خواهد ماند؛ علت فرض زیاد بودن بار کانال نیز همین موضوع است.

در پروتکل ۶ این مشکل نیز برطرف شده است. وقتی یک فریم (که طبق ترتیب مورد انتظار فرستاده شده) از راه رسید، گیرنده یک تایمر کمکی را (با تابع *start_ack_timer*) راه می‌اندازد. اگر در مدتی که این تایمر منقضی می‌شود، فریمی برای ارسال تحویل لایهٔ پیوند داده نشد، یک فریم تصدیق دریافت مستقل برگردانده خواهد شد (رویداد این تایمر *ack_timeout* نام دارد). با این تمهید دیگر نیازی نیست متکی به ترافیک سنگین دوطرفه باشیم، و پروتکل ۶ حتی می‌تواند بصورت کاملاً یکطرفه هم کار کند. از این تایمر کمکی فقط یکی وجود دارد، و اجرای تابع *start_ack_timer* آنرا ریست می‌کند. البته ضروری است که فاصلهٔ زمانی تایمر کمکی بسیار کوتاهتر از فاصلهٔ زمانی تایمرهای فریمهای داده باشد، تا این تایمرها قبل از رسیدن تصدیق دریافت منقضی نشوند.

استراتژی مقابله با خطا در پروتکل ۶ بسیار کارآمدتر از پروتکل ۵ است. اگر گیرنده به هر دلیلی ظن خطا ببرد، یک فریم **تصدیق دریافت منفی** (NAK) به فرستنده پس می‌فرستد. این NAK صریحاً از فرستنده می‌خواهد که فریم مشخص شده را دوباره ارسال کند. دو حالت وجود دارد که گیرنده باید به بروز خطا مشکوک شود: دریافت یک فریم معیوب، یا دریافت فریمی که انتظار آنرا ندارد. برای اجتناب از تکرار این درخواستها، گیرنده لیستی از فریمهایی که برای آنها NAK فرستاده را نگه می‌دارد. اگر متغیر *no_nak* در پروتکل ۶ مقدار true داشته باشد، نشان می‌دهد که هنوز برای فریم *frame_expected* هیچ NAK فرستاده نشده است. خراب یا گم شدن NAK ها مشکلی بوجود نخواهد آورد، فقط زمان ارسال مجدد فریمها را کمی به تأخیر می‌اندازد (چون تایمر فرستنده بهر حال منقضی خواهد شد). اگر پس از NAK فریم خواسته شده از راه برسد، گیرنده مقدار *no_nak* را به true ست کرده و تایمر کمکی را راه می‌اندازد (*start_ack_timer*). پس از انقضای این تایمر، گیرنده یک فریم ACK بـه فرستنده پس می‌فرستد، و بدین‌ترتیب خود را با آن سنکرون می‌کند.

در برخی شرایط، زمان لازم برای سفر فریمهای داده به مقصد، پردازش در آنجا، و بازگشت فریم تـصدیق

دریافت (تقریباً) ثابت است. در چنین شرایطی فرستنده می‌تواند تایمرهای خود را کمی بالاتر از این مقدار تنظیم کند. اما اگر این زمان در حد وسیعی متغیر باشد، فرستنده دو راه در پیش رو دارد: فاصلهٔ زمانی تایمرهای خود را خیلی کوچک بگیرد (و ریسک ارسالهای تکراری را بپذیرد)، یا آنرا بسیار بزرگ بگیرد (و بعد از هر خطا مدت زیادی بیکار بماند). هر دوی این گزینه‌ها تلف کردن پهنای باند است.

اگر ترافیک جهت مخالف کم باشد، زمان برگشت فریمهای تصدیق دریافت نیز نامنظم خواهد بود (گاهی کم و گاهی زیاد). تغییر زمان پردازش فریم در گیرنده هم می‌تواند مزید علت باشد. در کل، اگر انحراف معیار فاصلهٔ زمانی فریم تصدیق دریافت (در مقایسه با کل فاصلهٔ زمانی) کوچک باشد، می‌توان فاصلهٔ زمانی تایمرها را «کوچک» در نظر گرفت، که در این حالت NAK سودمندی خود را از دست می‌دهد. در غیر اینصورت، برای اجتناب از تکرار در ارسال فریمها باید فاصلهٔ زمانی تایمرها را «بزرگ» گرفت؛ در این حالت استفاده از NAK می‌تواند ارسال مجدد فریمهای معیوب و گم شده را تسریع کند.

سؤال دیگری که در همین زمینه پیش می‌آید این است که: کدام فریم باعث انقضای تایمر شده است؟ پروتکل ۵ همیشه *ack_expected* است (یعنی فقط انتظار دریافت تصدیق را دارد)، چون همیشه قدیمی‌ترین باعث انقضای تایمر می‌شود. اما در پروتکل ۶ راه ساده‌ای برای تعیین این موضوع وجود ندارد. فرض کنید فریمهای 0 تا 4 فرستاده شده‌اند، و لیست فریمهای بافر شده در فرستنده از قدیم‌ترین به جدیدترین (از چپ براست) عبارتند از: 01234 . حال وضعیت زیر را در نظر بگیرید: فریم 0 منقضی می‌شود، فرستنده فریم (جدید) 5 را می‌فرستد، فریم 1 منقضی می‌شود، و فرستنده فریم (جدید) 6 را می‌فرستد. در این لحظه، بافر فرستنده حاوی فریمهای 3405126 (از قدیم به جدید) است. اگر در این لحظه تمام فریمهای برگشتی (که فرض می‌کنیم حامل فریمهای تصدیق دریافت هستند) از بین بروند، هفت فریمی که در بافر قرار دارند، به همین ترتیب منقضی می‌شوند.

برای اجتناب از پیچیدگی بیشتر (که تا همین جا هم باندازهٔ کافی پیچیده هست)، به مدیریت تایمرها نپرداختیم. بجای آن فرض کردیم که متغیر *oldest_frame* در لحظهٔ انقضای تایمر نشان می‌دهد که کدام فریم منقضی شده است.

۵ـ۳ ارزیابی پروتکل‌ها

پروتکلهای واقعی (و کُد نرم‌افزاری آنها) اغلب بسیار پیچیده‌اند، به همین دلیل تحقیقات زیادی انجام شده تا روشهای ریاضی و مشخصی برای ارزیابی آنها ابداع شود. در این قسمت برخی از این تکنیکها و مدلها را بررسی خواهیم کرد. با اینکه در اینجا برای ارزیابی پروتکلهای لایهٔ پیوند داده از این تکنیکها استفاده کرده‌ایم، ولی آنها را می‌توان برای لایه‌های دیگر نیز بکار گرفت.

۵ـ۳ـ۱ مدل ماشین حالت محدود

یکی از مفاهیم کلیدی در مدلسازی پروتکلها، **ماشین حالت محدود** (finite state machine) است. در این تکنیک، هر **ماشین پروتکل** (protocol machine) – یعنی، فرستنده یا گیرنده – در هر لحظه از زمان در حالتی خاص قرار دارد. این **حالت** (state) عبارتست از مقدار تمام متغیرها، و **شمارندهٔ برنامه** (program counter) .

در اکثر مواقع می‌توان تعداد زیادی از حالت‌ها را برای آنالیز دسته‌بندی کرد. برای مثال، گیرندهٔ پروتکل ۳ را در نظر بگیرید؛ تمام حالت‌های ممکن این گیرنده را می‌توان به دو دستهٔ مهم تقسیم کرد: انتظار برای فریم 0 ، و انتظار برای فریم 1 . تمام حالت‌های دیگر را می‌توان مراحل گذار از یکی از این دو حالت به حالت دیگر دانست. معمولاً حالت‌ها را بگونه‌ای انتخاب می‌کنند که در آن لحظه ماشین پروتکل در انتظار وقوع رویداد بعدی است (در مثال ما، اجرای روال *wait(event)*). در این لحظه، حالت پروتکل را می‌توان بطور کامل با دانستن مقدار متغیرهای آن

تعیین کرد. تعدادی حالت‌های چنین ماشینی 2^n است، که در آن n تعداد بیت‌های لازم برای نمایش تمام ترکیبات ممکنهٔ متغیرهای آن است.

حالت کل سیستم نیز عبارتست از ترکیب حالت‌های دو ماشین پروتکل (فرستنده و گیرنده) و حالت کانال. حالت کانال با محتویات آن تعیین می‌شود. اگر باز هم از پروتکل ۳ کمک بگیریم، کانال ما می‌تواند حالتهای زیر را داشته باشد: فریم 0 یا 1 از فرستنده به گیرنده می‌رود، فریم تصدیق دریافت از گیرنده به فرستنده برمی‌گردد، و یا کانال خالیست. اگر فرض کنیم گیرنده و فرستنده نیز هر کدام فقط دو حالت دارند، کل سیستم دارای ۱۶ حالت مجزا خواهد بود.

همین جا باید نکتهٔ کوچکی را دربارهٔ کانال توضیح دهیم. وقتی می‌گوئیم «فریم در کانال است»، البته از یک مفهوم مجرد حرف می‌زنیم. آنچه واقعاً منظور ماست، اینست که «فریم احتمالاً به مقصد رسیده، ولی هنوز پردازش نشده‌است». این فریم تا زمانی که ماشین پروتکل روال *FromPhysicalLayer* را اجرا نکرده و آنرا پردازش نکند، «در کانال می‌ماند».

هر حالت دارای تعدادی **گذار** (transition) به حالتهای دیگر است. گذار زمانی روی می‌دهد که رویدادی رخ دهد. در یک ماشین پروتکل، ارسال یک فریم، دریافت یک فریم، انقضای یک تایمر و مانند آن، همگی نمونه‌هایی از گذار هستند. رویدادهایی که می‌توانند در یک کانال رخ دهد، نیز عبارتند از: وارد شدن یک فریم جدید به کانال توسط ماشین پروتکل، برداشته شدن فریم از کانال توسط ماشین پروتکل، و یا گم شدن فریم در اثر نویز. با در دست داشتن توصیف کاملی از ماشینهای پروتکل و مشخصات کانال، می‌توان سیستم را بصورت نموداری از گره‌ها (حالت‌ها) و خطوط متصل‌کننده (گذارها) نمایش داد.

یکی از حالت‌های مهم در هر سیستم، **حالت اولیه** (initial state) است. این حالت متناظر است با وضعیت سیستم در لحظهٔ شروع به کار، یا (اگر مناسبتر باشد) کمی پس از آن. از این حالت اولیه می‌توان به کمک توالی گذارها به تمام (یا برخی از) حالت‌های دیگر رسید. با کمک تکنیکهای **نظریهٔ گراف** (graph theory) می‌توان مشخص کرد که کدام حالت‌ها قابل دسترسی‌اند، و کدامها نیستند. با این تکنیک، که به **آنالیز دسترسی** (reachability analysis) معروفست (Lin et al., 1987)، می‌توان دریافت که آیا یک پروتکل درست عمل می‌کند یا خیر.

مدل ماشین حالت محدود یک پروتکل را می‌توان بردار چهار عضوی (S, M, I, T) دانست، که در آن:

۱. S عبارتست از مجموعهٔ حالت‌هایی که پروسس‌ها و کانال می‌توانند در آن باشند.
۲. M عبارتست از مجموعهٔ فریمهایی که می‌توان روی کانال مبادله کرد.
۳. I عبارتست از مجموعهٔ حالت‌های اولیهٔ پروسس‌ها.
٤. T عبارتست از مجموعهٔ گذارهای بین حالت‌ها.

در لحظهٔ شروع، تمام پروسسها در حالت اولیه‌شان هستند. سپس رویدادها شروع به رخ دادن می‌کنند: فریمی برای ارسال آماده می‌شود، تایمرها خاموش می‌شوند، و مانند آن. هر رویداد می‌تواند باعث شود که یک پروسس یا کانال عملی را انجام داده و به حالت دیگر برود. با تعیین دقیق پیامدهای هر حالت، می‌توان گراف دسترسی را رسم و پروتکل را آنالیز کرد.

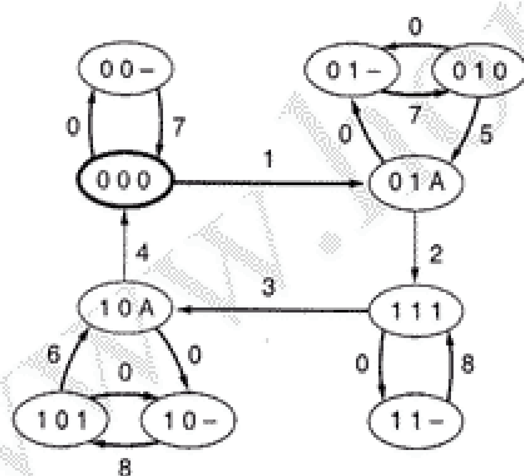
آنالیز دسترسی می‌تواند خطاهای مختلفی را در طراحی پروتکل روشن کند. برای مثال، اگر یک فریم خاص بتواند در حالت خاصی وجود داشته باشد و ماشین حالت محدود نتواند بگوید در این موقعیت چه باید کرد، طراحی ما مشکل دارد (ناقص است). اگر حالت یا حالتهایی وجود داشته باشد که نتوان از آن خارج شد (بعبارت دیگر، دریافت فریم سالم دیگر امکانپذیر نباشد)، باز هم پروتکل ما مشکل دارد (بن‌بست). موقعیت دیگری (که

در واقع مشکل چندان بزرگی نیست) آنست که پروتکل ما برای حالتهایی تجهیز شده باشد که امکان رُوی دادن آنها وجود ندارد (گذارهای نامربوط). گراف دسترسی خطاهای دیگر را هم می‌تواند کشف کند.

در شکل ۳-۲۱ (الف) نمونه‌ای از یک مدل ماشین حالت محدود را ملاحظه می‌کنید. این گراف معادل پروتکل ۳ است، که در بالا توضیح دادیم: هر ماشین پروتکل دارای دو حالت، و کانال دارای چهار حالت است. در کل ۱۶ حالت ممکن وجود دارد، که البته از حالت اولیه نمی‌توان به همهٔ آنها رسید. در شکل این حالتهای غیر قابل دسترسی را نشان نداده‌ایم، و برای سادگی کار از خطاهای **جمع تطبیقی** (checksum) هم چشم پوشیده‌ایم.

هر حالت با سه حرف *SRC* مشخص می‌شود، که در آن *S* یا 0 است یا 1 (متناظر با فریمی که فرستنده می‌خواهد بفرستد)؛ *R* نیز یا 0 است یا 1 (متناظر با فریمی که گیرنده منتظر دریافت آن است)؛ و *C* می‌تواند چهار مقدار 0 (فریم 0)، 1 (فریم 1)، A (فریم تصدیق دریافت) یا خالی (-) بگیرد (متناظر با حالتهای چهارگانهٔ کانال). در مثال بالا، حالت اولیه با (000) نشان داده شده است: یعنی فرستنده فریم 0 را فرستاده، گیرنده منتظر دریافت فریم 0 است، و فریم 0 اکنون در کانال است.

در شکل ۳-۲۱ نُه حالت گذار مختلف نشان داده شده است. در گذار 0 کانال محتویات خود را از دست می‌دهد. در گذار 1 کانال محتویات خود را به گیرنده تحویل می‌دهد، و گیرنده نیز حالت خود را به 1 (انتظار برای فریم 1) تغییر داده و یک فریم تصدیق دریافت به فرستنده پس می‌فرستد. گذار 1 همچنین متناظرست با تحویل بستهٔ 0 به لایهٔ شبکه در گیرنده. گذارهای دیگر را در شکل ۳-۲۱ (ب) ملاحظه می‌کنید. رسیدن فریمی با خطای جمع تطبیقی در اینجا نشان داه نشده، چون این اتفاق در پروتکل ۳ باعث تغییر حالت نمی‌شود.



(الف)

انتقال	نوبت کیست؟	فریم پذیرفته شده	فریم ارسال شده	به لایه شبکه
0	–	(فریم گم شده)		–
1	R	0	A	Yes
2	S	A	1	–
3	R	1	A	Yes
4	S	A	0	–
5	R	0	A	No
6	R	1	A	No
7	S	(انقضای زمان)	0	–
8	S	(انقضای زمان)	1	–

(ب)

شکل ۳-۲۱. (الف) دیاگرام حالت پروتکل ۳. (ب) گذارها.

در حالت عادی، گذارهای 1، 2، 3 و 4 پشت سرهم و بارها و بارها تکرار می‌شوند. در هر سیکل دو بسته منتقل می‌شود، و با این کار فرستنده دوباره به حالت اولیه (ارسال فریم 0) برمی‌گردد. اگر فریم 0 در کانال از بین برود، سیستم از حالت (000) به حالت (-00) می‌رود (گذار 0). پس از مدتی تایمر فرستنده به انتها رسیده (گذار 7)، و سیستم به حالت (000) بازمی‌گردد. از بین رفتن فریم A (تصدیق دریافت) پیچیده‌تر است، و برای جبران آن به دو گذار نیاز داریم: 7 و 5، یا 6 و 8.

یکی از ویژگیهایی که یک پروتکل با شماره‌های ترتیبی 1-بیتی باید داشته باشد اینست که هرگز نباید دو فریم فرد متوالی (یا دو فریم زوج متوالی) به گیرنده برسد. در شکل ۳-۲۱ می‌توان این ویژگی را چنین نشان داد: «هیچ

مسیری از حالت اولیه وجود ندارد که طی آن دو گذار متوالی 1 رخ دهد، بدون اینکه بین آنها یک گذار 3 وجود داشته باشد، و بالعکس.» از شکل می‌توان دید که پروتکل ۳ این ویژگی را دارد.

الزام دیگر اینست که هیچ مسیری نباید وجود داشته باشد که طی آن فرستنده دو بار تغییر حالت دهد (از 0 به 1 و برگشت دوباره به 0) در حالیکه گیرنده ثابت مانده است. اگر چنین مسیری وجود داشته باشد، بمعنای آن است که دو فریم از بین رفته بدون اینکه گیرنده متوجه شده باشد.

ویژگی مهمتر یک پروتکل عدم وجود **بن‌بست** (deadlock) در آن است. بن‌بست حالتی است که پروتکل (تحت هیچ شرایطی) دیگر قادر به جلو رفتن نباشد (یعنی نتواند بسته‌ها را لایهٔ شبکه تحویل دهد). در مدل گراف، بن‌بست به زیرمجموعه‌ای از حالتها گفته می‌شود که بتوان از شرایط اولیه به آن رسید و دو ویژگی زیر را داشته باشد:

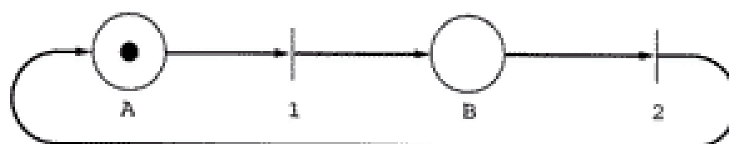
۱. هیچ گذاری برای خروج از این زیرمجموعه وجود نداشته باشد.
۲. هیچ گذاری در داخل زیرمجموعه وجود نداشته باشد که باعث پیشرفت کار شود.

وقتی یک پروتکل وارد بن‌بست شود، دیگر برای همیشه آنجا می‌ماند. باز هم از گراف شکل ۳-۲۱ می‌توان دید که (خوشبختانه) پروتکل ۳ هیچ بن‌بستی ندارد.

۳-۵-۲ مدل شبکهٔ پتری

ماشین حالت محدود تنها مدل برای ارزیابی پروتکلها نیست. در این قسمت به بررسی تکنیکی کاملاً متفاوت بنام **شبکهٔ پتری** (Petri net) می‌پردازیم (Danthine, 1980). یک شبکهٔ پتری دارای چهار عنصر اساسی است: **مکان** (place)، **گذار** (transition)، **کمان** (arc)، و **نشانه** (token). مکان حالتیست که سیستم (یا بخشی از آن) می‌تواند در آن باشد. در شکل ۳-۲۲ یک شبکهٔ پتری را با دو مکان A و B می‌بینید، که با دایره مشخص شده‌اند. سیستم در حال حاضر در حالت A قرار دارد، که این موضوع با نشانه (نقطهٔ سیاه) در مکان A مشخص شده است. گذار با یک خط افقی یا عمودی مشخص می‌شود. هر گذار می‌تواند دارای تعدادی **کمان ورودی** (input arc - که از مکانهای ورودی آن می‌آیند) و تعدادی **کمان خروجی** (output arc - که به مکانهای خروجی آن می‌روند) باشد.

گذار **فعال** به گذاری گفته می‌شود که حداقل یک نشانه در یکی از ورودیهای آن وجود داشته باشد. گذار فعال می‌تواند هر گاه که اراده کند، **آتش** (fire) کرده و یک نشانه را از یکی از ورودیها برداشته و در یکی از خروجیهای خود قرار دهد. اگر تعداد کمانهای ورودی با کمانهای خروجی مساوی نباشد، نشانه **ابقا** (conserve) نخواهد شد.

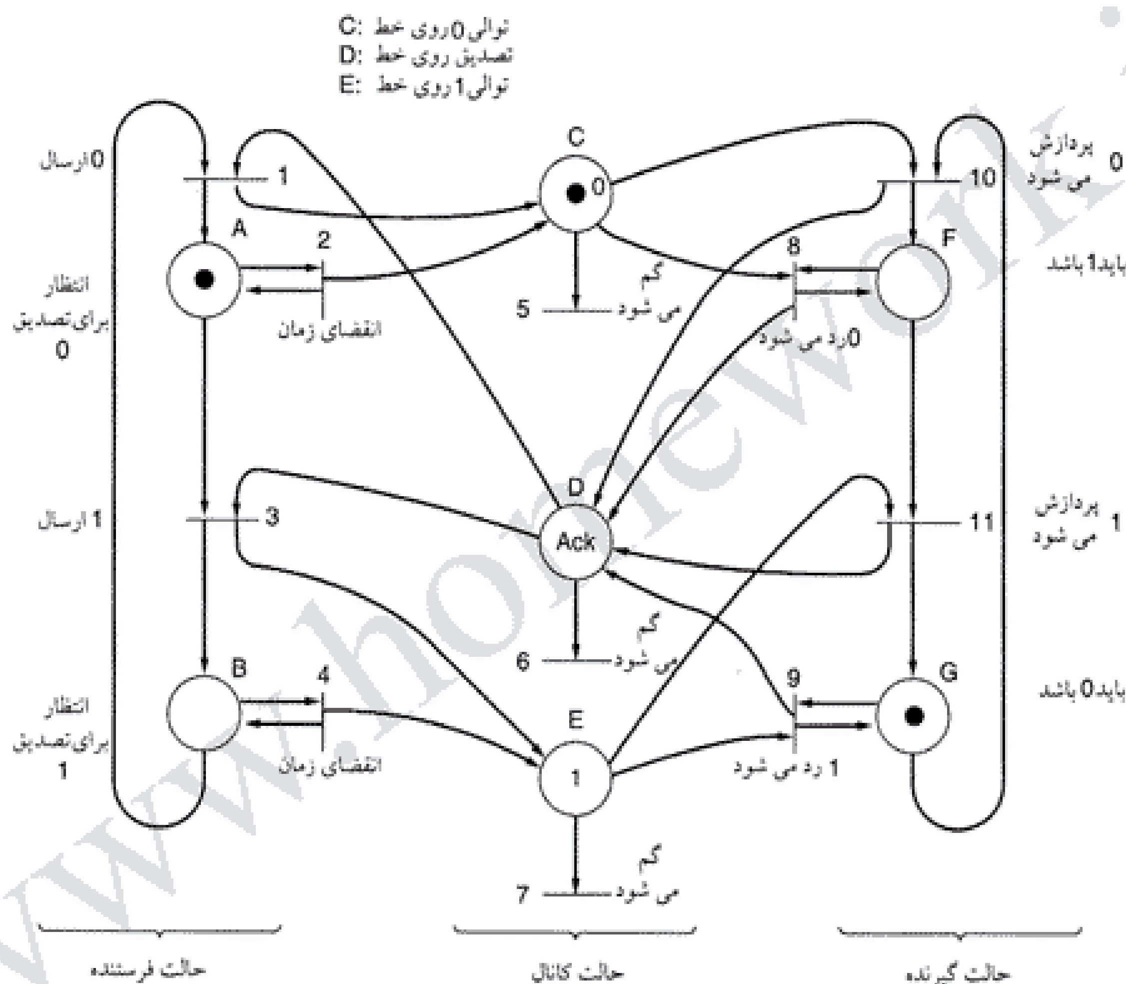


شکل ۳-۲۲. یک شبکهٔ پتری با دو مکان و دو گذار.

اگر دو یا چند گذار فعال وجود داشته باشد، هر کدام از آنها می‌توانند آتش کنند. اینکه کدام گذار آتش می‌کند نامشخص است، و همین ویژگیست که شبکهٔ پتری را برای مدلسازی پروتکل‌ها سودمند کرده است. شبکهٔ پتری شکل ۳-۲۲ کاملاً مشخص است و از آن فقط برای مدلسازی پروسسهایی که دو فاز بیشتر ندارند، می‌توان استفاده کرد (مانند رفتار یک نوزاد: خوردن، خوابیدن، خوردن، خوابیدن، و الی‌آخر). در اینجا هم مانند سایر تکنیکهای مدلسازی جزئیات زائد حذف می‌شوند.

در شکل ۳-۲۳ مدل شبکهٔ پتری شکل ۳-۱۲ (پروتکل ۳) را ملاحظه می‌کنید. بر خلاف مدل ماشین حالت

محدود، در اینجا حالتهای ترکیبی وجود ندارد؛ حالت فرستنده، گیرنده و کانال بطور مجزا و مستقل نمایش داده می‌شوند. گذارهای 1 و 2 بترتیب عبارتند از ارسال فریم 0 توسط فرستنده در حالت عادی، و بعد از انقضای تایمر. گذارهای 3 و 4 متناظر با ارسال فریم 1 در این دو موقعیت هستند. گذارهای 5 ، 6 و 7 نیز بترتیب از بین رفتن فریمهای 0 ، ACK و 1 را نشان می‌دهند. گذارهای 8 و 9 نشان می‌دهند که فریمی با شمارهٔ ترتیبی اشتباه (بترتیب 0 و 1) به گیرنده رسیده است. گذارهای 10 و 11 نیز بترتیب حاکی از رسیدن صحیح و سالم فریمهای 0 و 1 به گیرنده، و تحویل آنها به لایهٔ شبکه هستند.



شکل ۳-۲۳. مدل شبکهٔ پتری پروتکل ۳.

با شبکهٔ پتری نیز می‌توان مانند ماشین حالت محدود مشکلات یک پروتکل را تشخیص داد. برای مثال، اگر یک توالی آتش وجود داشته باشد که در آن دو گذار 10 (بدون یک گذار 11 بین آنها) رخ دهد، پروتکل ما مشکل دارد. مفهوم بن‌بست در شبکهٔ پتری نیز کاملاً شبیه همین مفهوم در ماشین حالت محدود است.

شبکهٔ پتری را بصورت جبری نیز می‌توان نوشت: هر گذار یک جملهٔ جبری است، که در یک دستور (که مکانهای ورودی و خروجی گذار را مشخص می‌کند) بکار می‌رود. از آنجائیکه شکل ۳-۲۳ دارای ۱۱ گذار است، برای نمایش جبری آن به ۱۱ دستور (که آنها را متناظر با گذارها شماره‌گذاری می‌کنیم) نیاز داریم. معادل جبری شبکهٔ پتری شکل ۳-۲۳ چنین است:

1: BD → AC
2: A → A
3: AD → BE
4: B → B
5: C →
6: D →
7: E →
8: CF → DF
9: EG → DG
10: CG → DF
11: EF → DG

همانطور که می‌بینید، یک پروتکل نسبتاً پیچیده به ۱۱ جملهٔ جبری ساده تبدیل شده که به آسانی می‌توان آنرا با کامپیوتر تحلیل کرد.

حالت فعلی شبکهٔ پتری با مجموعهٔ نامنظمی از مکانها (که هر مکان به تعداد نشانه‌هایی که دارد، ظاهر می‌شود) نشان داده می‌شود. در هر دستور، مکانهایی که سمت چپ جمله قرار دارند می‌توانند آتش کرده، و (بعد از حذف خود از حالت فعلی) خروجی خود را به حالت سیستم اضافه کنند. شکل ۳-۲۳ را می‌توان با علامت *ACG* نشان داد (یعنی، مکانهای *A*، *C* و *G* هر کدام یک نشانه دارند). در نتیجه، دستورات 2 ، 5 و 10 فعال هستند، و هر کدام از آنها می‌توانند اجرا شوند و حالت سیستم را عوض کنند (که البته حالت قبلی هم جزء حالتهای ممکنه است). توجه داشته باشید که، برای مثال، در این لحظه دستور 3 (AD → BE) نمی‌تواند اجرا شود، چون *D* در علامت سیستم وجود ندارد.

۶ـ۳ چند نمونه از پروتکلهای لینک داده

در این قسمت چند نمونه از پروتکلهای لینک داده را که کاربرد وسیعی دارند، مورد بررسی قرار خواهیم داد. اولین آنها، HDLC ، یک پروتکل بیت-گرا (bit-oriented) است که سالهاست در برنامه‌های بسیاری از ویرایشهای مختلف آن استفاده می‌شود. دومی، PPP ، یک پروتکل لینک داده است که برای اتصال کامپیوترهای خانگی به اینترنت بکار می‌رود.

۳ـ۶ـ۱ HDLC - کنترل سطح بالای لینک داده

در این قسمت گروهی از پروتکلهای نزدیک به هم را بررسی می‌کنیم که با وجود قدیمی بودن، همچنان کاربرد گسترده‌ای دارند. همهٔ این پروتکلها از اولین پروتکل لینک داده که برای کامپیوترهای بزرگ IBM توسعه داده شد، مشتق شده‌اند: پروتکل SDLC (**کنترل لینک دادهٔ سنکرون** - Synchronous Data Link Control). بعد از توسعهٔ این پروتکل، IBM آنرا برای پذیرش بعنوان استاندارد آمریکایی و بین‌المللی به ANSI و ISO فرستاد. ANSI و ISO هر کدام تغییراتی در این پروتکل دادند، و بترتیب پروتکلهای ADCCP (**روش پیشرفتهٔ کنترل مخابرات داده** - Advanced Data Communication Control Procedure) و HDLC (**کنترل سطح بالای لینک داده** - High-Level Data Link Control) را از آن مشتق کردند. بعد از مدتی، CCITT تغییراتی در پروتکل HDLC داده، و پروتکل جدید را که LAP (**روش دسترسی لینک** - Link Access Procedure) نام گرفت، بعنوان قسمتی از استاندارد شبکه‌های X.25 معرفی کرد (این پروتکل بعدها برای سازگاری بهتر با ویرایشهای جدید HDLC باز هم اصلاح و LAPB نامیده شد). خوبی استانداردها همین تنوع زیاد آنهاست، و بالاخره می‌توانید یکی را مطابق سلیقه‌تان پیدا کنید؛ اگر هم پیدا نکردید، زیاد جای نگرانی نیست: سال آینده

مدلهای جدیدتری به بازار خواهد آمد!

تمام این پروتکلها مبنای واحدی دارند: همهٔ آنها بیت-گرا هستند، و از تکنیکهای **لاگذاری بیت** (bit stuffing) برای افزونگی داده‌ها استفاده می‌کنند. اختلاف آنها کوچک (ولی بهر حال، ناراحت‌کننده) است. برای اطلاع از مشخصات دقیق هر پروتکل می‌توانید به منابع مربوطه مراجعه کنید.

تمام پروتکلهای بیت-گرا از فریمهایی با ساختار شکل ۳-۲۴ استفاده می‌کنند. هر فریم با یک توالی پرچم (01111110) شروع می‌شود. فیلد **آدرس** (Address) در خطوطی اهمیت می‌یابد که ترمینالهای متعددی دارند، و از این فیلد برای مشخص کردن ترمینال مقصد استفاده می‌شود. در خطوط **نقطه-به-نقطه** (point-to-point) گاهی از این فیلد برای تشخیص **فرمان** (command) از **پاسخ** (response) استفاده می‌شود.

فیلد **کنترل** (Control) برای شمارهٔ ترتیبی فریم، تصدیق دریافت، و مقاصد دیگر بکار می‌رود (در ادامه این فیلد را بیشتر توضیح خواهیم داد).

Bits (بیت‌ها)	8	8	8	≥ 0	16	8
	01111110	Address (آدرس)	Control (کنترل)	Data (داده)	Checksum (جمع تطبیقی)	01111110

شکل ۳-۲۴. فرمت فریم در پروتکلهای بیت-گرا.

فیلد **داده** (Data) محتوی اطلاعاتیست که فریم باید منتقل کند. طول این فیلد می‌تواند هر اندازه‌ای باشد، اگر چه با زیاد شدن آن کارایی تکنیک جمع تطبیقی (بدلیل بالا رفتن احتمال بروز خطاهای فورانی) کاهش خواهد یافت.

فیلد **جمع تطبیقی** (Checksum) یک کُد **افزونگی چرخه‌ای** (cyclic redundancy) است، که در قسمت ۳-۲-۲ توضیح دادیم.

در انتها، فریم به یک توالی پرچم دیگر (01111110) ختم می‌شود. وقتی یک خط نقطه-به-نقطه بیکار است، بطور پیوسته توالیهای پرچم را ارسال می‌کند. هر فریم باید حداقل سه فیلد (مجموعاً ۳۲ بیت) داشته باشد (البته منهای پرچمهای ابتدا و انتها).

فریمها بر سه نوعند: **اطلاعاتی** (Information)، **سرپرستی** (Supervisory)، و **بدون‌شماره** (Unnumbered).

در شکل ۳-۲۵ فیلد کنترل هر یک از این سه نوع فریم را ملاحظه می‌کنید. این پروتکل از تکنیک پنجرهٔ لغزنده، با شماره‌های ترتیبی ۳-بیتی، استفاده می‌کند. در هر لحظه تا هفت فریم تصدیق نشده می‌تواند در بافر فرستنده وجود داشته باشد. فیلد *Seq* در شکل ۳-۲۵ (الف) شمارهٔ ترتیبی فریم را نشان می‌دهد. فیلد *Next* نیز فیلد سواری مجانی برای تصدیق دریافت است. با این حال، در تمام انوع پروتکلهای HDLC مرسوم است که بجای سوار کردن شمارهٔ آخرین فریم دریافت شده، شمارهٔ اولین فریمی که هنوز دریافت نشده (فریمی که گیرنده منتظر آن است) برگردانده شود. این دو روش هیچ مزیتی بر یکدیگر ندارند، و انتخاب یکی از آنها به طراح پروتکل بستگی دارد؛ البته مشروط باینکه همواره از یک روش استفاده کند.

فیلد *P/F* مخفف *Poll/Final* (سرکشی/پایان) است. این فیلد در کامپیوترها یا مودمهایی بکار می‌رود که به چندین ترمینال **سرکشی** (polling) می‌کنند. اگر این فیلد محتوی *P* باشد، کامپیوتر (یا مودم) ترمینال را دعوت به ارسال داده می‌کند. در تمام فریمهایی که ترمینال می‌فرستد، فیلد *P/F* مقدار *P* دارد (بجز در آخرین فریم، که مقدار آن *F* است).

در برخی از پروتکلها بیت *P/F* باعث می‌شود تا ماشین طرف مقابل بلافاصله فریم سرپرستی را بفرستد، و منتظر سواری مجانی نشود. در ارتباطاتی که از فریمهای بدون‌شماره سود می‌برند، نیز این بیت کاربرد دارد.

Bits	1	3		1	3
(الف)	0	Seq(توالی)		P/F	Next(بعدی)
(ب)	1	0	Type(نوع)	P/F	Next(بعدی)
(ج)	1	1	Type(نوع)	P/F	Modifier(تغییردهنده)

شکل ۳-۲۵. فیلد کنترل در یک (الف) فریم اطلاعاتی، (ب) فریم سرپرستی، (ج) فریم بدون‌شماره.

انواع مختلف فریمهای سرپرستی با فیلد *Type* مشخص می‌شود. نوع 0 فریم تصدیق دریافت (که رسماً RECEIVE READY نامیده می‌شود) است، و مشخص می‌کند که گیرنده آمادهٔ دریافت فریم بعدیست. این فریم وقتی ارسال می‌شود که ترافیک جهت برگشت برای اجرای تکنیک سواری مجانی وجود نداشته باشد.

نوع 1 فریم تصدیق دریافت منفی (که رسماً REJECT نامیده می‌شود) است، ، از آن برای تصدیق دریافت فریم با خطا استفاده می‌شود. در این حالت فیلد *Next* حاوی اولین فریمیست که درست دریافت نشده است (یعنی فریمی که باید دوباره ارسال شود). در اینجا فرستنده باید تمام فریمهای بعد از فریم *Next* را مجدداً بفرستد، و از این نظر شبیه پروتکل 5 است تا پروتکل 6 .

نوع 2 فریم RECEIVE NOT READY است؛ این فریم اعلام می‌کند که تمام فریمهای قبل از *Next* درست دریافت شده‌اند، ولی خود *Next* خیر (و از این نظر شبیه RECEIVE READY است، با این تفاوت که جلوی ادامهٔ ارسال را می‌گیرد). فریم RECEIVE NOT READY در واقع نوعی اعلام مشکل است از سوی گیرنده، مثلاً مشکل پُر شدن بافرها. بعد از برطرف شدن این وضعیت، گیرنده یکی از فریمهای کنترلی دیگر (مانند RECEIVE READY یا REJECT) را می‌فرستد.

نوع 3 فریم SELECTIVE REJECT است. این فریم فقط زمانی فرستاده می‌شود که گیرنده خواستار ارسال مجدد یک فریم خاص باشد؛ HDLC از این نظر شبیه پروتکل 6 است تا پروتکل 5 ، و برای مواقعی مفید است که اندازهٔ پنجرهٔ ارسال نصف شمارهٔ ترتیبی باشد. بنابراین اگر گیرنده بخواهد فریمهای نامنظم را بافر کرده و فقط ارسال مجدد برخی از فریمهای معیوب را طلب کند، می‌تواند از SELECTIVE REJECT استفاده کند. این فریم کنترلی فقط در HDLC و ADCCP وجود دارد، ولی در SDLC و LAPB تعریف نشده است.

سومین نوع از فریمها، فریمهای **بدون‌شماره** (Unnumbered) است. این نوع گاهی کاربردهای کنترلی دارد، ولی در سرویسهای غیرقابل‌اعتماد غیرمتصل نیز می‌توان از آن برای انتقال داده استفاده کرد. برخلاف دو نوع دیگر، این نوع فریم عملکردهای متفاوتی در انواع پروتکلهای بیت-گرا است. در این حالت پنج بیت برای تعیین کارکرد فریم وجود دارد، ولی تمام ۳۲ حالت آن استفاده نمی‌شود.

در تمام پروتکلها فرمانی وجود دارد بنام DISC (**قطع ارتباط**) که اجازه می‌دهد یک ماشین خاموش شدن خود را به ماشینهای دیگر اعلام کند. فرمان دیگری وجود دارد که به یک ماشین اجازه می‌دهد تا بازگشت خود را اعلام کرده، و تمام شماره‌های ترتیبی را به 0 برگرداند؛ این فرمان SNRM (**ست شدن حالت پاسخ عادی** - Set Normal Response Mode) نام دارد. متأسفانه، «حالت پاسخ عادی» هر چیزی هست جز عادی. این حالت عبارتست از یک رابطهٔ نامتقارن ، که در آن یک سر خط «ارباب» و سر دیگر خط «رعیت» است. این فرمان از زمانی

به جای مانده که یک کامپیوتر بزرگ مرکزی وجود داشت و تعداد زیادی ترمینال به آن متصل می‌شد، که در این حالت مسلماً رابطه نامتقارن ارباب و رعیتی «عادی» محسوب می‌شود. برای آن که این پروتکلها بتوانند پاسخگوی نیازهای جدید (رابطهٔ متقارن) باشند، در HDLC و LAPB فرمان دیگری وجود دارد بنام SABM (**ست شدن حالت آسنکرون متعادل** – Set Asynchronous Balanced Mode)، که خط را ریست کرده و دو سر آنرا به حالت هم‌ارز و متعادل درمی‌آورد. در این پروتکلها دو فرمان دیگر وجود دارد بنامهای SABME و SNRME که بترتیب با SABM و SNRM متناظر هستند، و فقط در آنها شمارهٔ ترتیبی فریمها (بجای ۳-بیتی) ۷-بیتی است.

فرمان سومی که در تمام پروتکلها وجود دارد، FRMR (FRaMe Reject) است که نشان می‌دهد جمع تطبیقی فریم صحیح است ولی از نظر شکلی درست نیست. از میان چنین شکلهای نادرستی می‌توان به فریم سرپرستی نوع 3 در LAPB ، فریمی که کوتاهتر از ۳۲ بیت است، و تصدیق دریافت فریمی که خارج از پنجرهٔ دریافت است، اشاره کرد. در فریم FRMR یک فیلد دادهٔ ۲۴ بیتی وجود دارد که علت خطا را توضیح می‌دهد. اطلاعاتی که در این فیلد مخابره می‌شود، عبارتند از: فیلد کنترل فریم معیوب، پارامترهای پنجرهٔ دریافت، و اطلاعاتی دربارهٔ طبیعت خطا.

احتمال خراب یا گم شدن فریمهای تصدیق دریافت نیز مانند سایر فریمها وجود دارد، بنابراین برای آنها نیز باید نوعی تصدیق دریافت پس فرستاد. برای این منظور فریم کنترلی خاصی بنام UA (**تصدیق دریافت بدون‌شماره** – Unnumbered Acknowledgement) در نظر گرفته شده است. از آنجائیکه همیشه فقط یک فریم تصدیق دریافت تصدیق نشده در گیرنده وجود دارد، هیچ ابهامی وجود ندارد که منظور از یک فریم UA کدام فریم تصدیق دریافت است.

سایر فریمهای کنترلی کارهایی مانند **آماده‌سازی** (initialization)، **سرکشی** (polling) و **گزارش وضعیت** (status report) انجام می‌دهند. فریم کنترلی دیگری نیز وجود دارد بنام UI (Unnumbered Information)، که می‌توان با آن اطلاعات دلخواه ارسال کرد؛ این اطلاعات برای لایهٔ پیوند داده هستند، و به لایهٔ شبکه داده نمی‌شوند.

پروتکل HDLC علیرغم کاربرد وسیع آن، بهیچوجه کامل نیست. در (Fiorini et al., 1994) می‌توانید بحثی دربارهٔ مشکلات این پروتکل را ببینید.

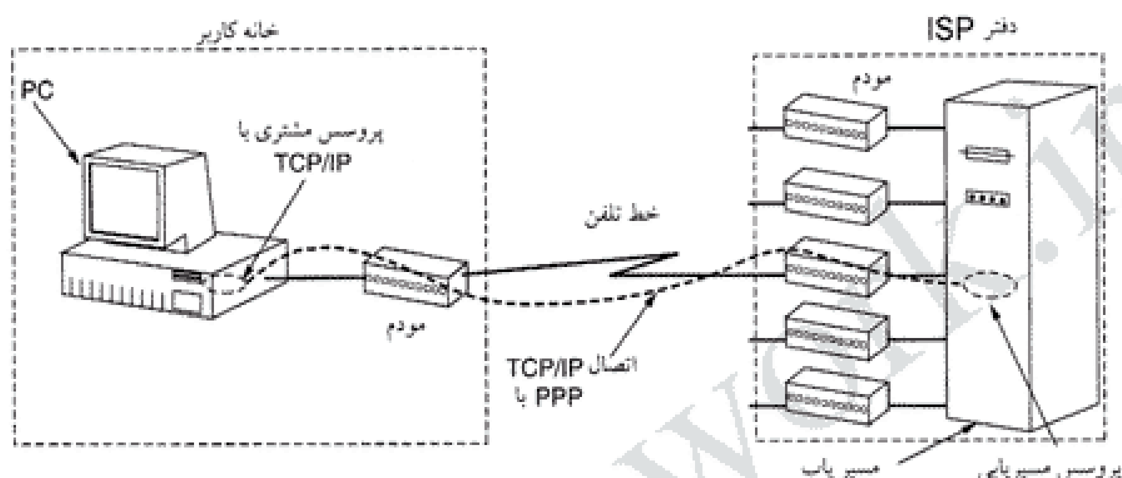
۳-۶-۲ لایهٔ پیوند داده در اینترنت

اینترنت شامل ماشینهای متعددی (میزبان و مسیریاب) است، که توسط یک ستون فقرات به یکدیگر متصل می‌شوند. در یک ساختمان کوچک می‌توان از تکنیکهای LAN برای ارتباط استفاده کرد، ولی در اینترنت اغلب ارتباطات از نوع **نقطه-به-نقطه** (piont-to-point) است. در فصل ۴ دربارهٔ لایهٔ پیوند داده در LAN صحبت خواهیم کرد؛ در این قسمت به لایهٔ پیوند داده در خطوط نقطه-به-نقطه می‌پردازیم.

در عمل، ارتباط نقطه-به-نقطه در دو حالت بکار برده می‌شود. اول، هزاران شرکت و مؤسسه دارای شبکه‌های محلی با تعداد زیادی ماشین میزبان (کامپیوترهای رومیزی، ایستگاههای کاری، کامپیوترهای سرویس‌دهنده و غیره) و یک مسیریاب (یا یک **پل** – bridge – که در عمل همان وظیفه را انجام می‌دهد) هستند، و این مسیریابها در یک شبکهٔ بزرگتر بیکدیگر متصل شده‌اند. معمولاً این مسیریابها بوسیلهٔ ارتباط نقطه-به-نقطه و از طریق **خطوط اجاره‌ای** (leased line) به یک (یا دو) مسیریاب دیگر متصل می‌شوند. همین مسیریابها و خطوط اجاره‌ای هستند که زیرشبکهٔ اینترنت را می‌سازند.

دومین حالتی که ارتباط نقطه-به-نقطه بکار برده می‌شود، میلیونها کاربر اینترنتی هستند که از منزل و با یک مودم (از طریق خط تلفن) به اینترنت متصل می‌شوند. اتفاقی که معمولاً می‌افتد اینست که کامپیوتر کاربر به

مسیریاب سرویس‌دهندهٔ اینترنت (ISP) زنگ می‌زند، و از آن طریق (درست مثل یک میزبان معمولی) به اینترنت متصل می‌شود. در این روش فرقی نمی‌کند که خط تلفن معمولی است یا اجاره‌ای، فقط بعد از اینکه کاربر دیگر نیازی به آن نداشت، ارتباط قطع می‌شود. در شکل ۳-۲۶ این نوع ارتباط نقطه-به-نقطه را ملاحظه می‌کنید. مودمی که در این شکل نشان داده‌ایم یک مودم خارجی است، اما مودمهای داخلی نیز دقیقاً همان کار را انجام می‌دهند.



شکل ۳-۲۶. یک کامپیوتر خانگی می‌تواند نقش میزبان اینترنت را بازی کند.

در هر دو حالت (ارتباط مسیریاب-مسیریاب یا میزبان-مسیریاب) به یک پروتکل لینک دادهٔ نقطه-به-نقطه نیاز داریم تا وظایفی از قبیل فریم‌بندی، کنترل خطا و مانند آن را انجام دهد. پروتکلی که در اینترنت از آن استفاده می‌شود (و در این قسمت آنرا بررسی خواهیم کرد)، PPP نام دارد.

PPP - پروتکل نقطه-به-نقطه

اینترنت در موارد مختلفی، از قبیل ترافیک مسیریاب-به-مسیریاب یا ترافیک کاربر-به-ISP، به پروتکل نقطه-به-نقطه نیاز دارد. این پروتکل PPP (**پروتکل نقطه-به-نقطه** - Point-to-Point Protocol) نام دارد، که در RFC 1661 تعریف شده و در چند RFC دیگر (از قبیل RFC 1662 و RFC 1663) مشخصات آن بهبود یافته است. PPP ویژگیهای کنترل خطا دارد، از پروتکلهای مختلف پشتیبانی می‌کند، اجازه می‌دهد تا آدرس IP در زمان اتصال به طرف مقابل درخواست شود، **تعیین هویت** (authentication) انجام می‌دهد، و دهها ویژگی دیگر. مهمترین بخشهای PPP عبارتند از:

۱. یک روش فریم‌بندی، که ابتدا و انتهای فریمها را بوضوح مشخص می‌کند. فرمت فریم در PPP تشخیص خطا را نیز انجام می‌دهد.

۲. یک پروتکل کنترل لینک برای برقراری ارتباط، تست آن، مذاکره برای سایر گزینه‌ها، و در پایان قطع ارتباط بصورتی آبرومندانه. این پروتکل LCP (پروتکل کنترل لینک - Link Control Protocol) نام دارد. LCP از هر دو ارتباط سنکرون و آسنکرون، و بیت-گرا یا بایت-گرا پشتیبانی می‌کند.

۳. روشی برای ارتباط و مذاکره دربارهٔ گزینه‌های لایهٔ شبکه، که از طرز کار این لایه مستقل است. در این روش برای هر نوع لایهٔ شبکه یک NCP (پروتکل کنترل شبکه - Network Control Protocol) وجود دارد.

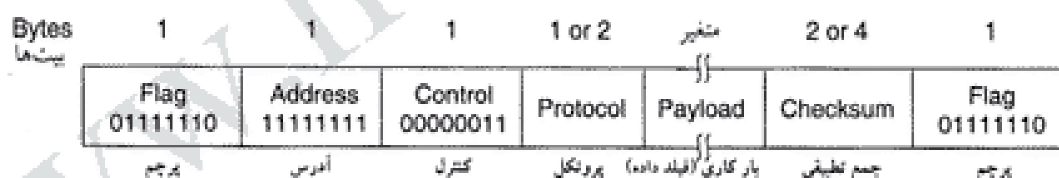
برای اینکه ببینید این قطعات چگونه با یکدیگر جفت می‌شوند، سناریوی ارتباط کاربر-به-ISP را در نظر می‌گیریم. ابتدا PC کاربر از طریق مودم خود به مسیریاب ISP زنگ می‌زند. بعد از اینکه مودم مسیریاب گوشی را

برداشت و ارتباط برقرار شد، PC از طریق فیلد دادهٔ یک یا چند فریم PPP چند بستهٔ LCP به مسیریاب می‌فرستد. پارامترهای PPP از طریق همین بسته‌ها و پاسخ آنها انتخاب می‌شود.

بعد از آنکه بر سر این پارامترها توافق حاصل شد، یک سری بستهٔ NCP برای پیکربندی لایهٔ شبکه رد و بدل می‌شود. معمولاً PC هایی که به اینترنت متصل می‌شوند از TCP/IP استفاده می‌کنند، پس PC ما به یک آدرس IP نیاز دارد. تعداد آدرسهای IP آنقدر زیاد نیست که بتوان به همه آدرس ثابت اختصاص داد، بهمین دلیل ISP تعدادی آدرس IP را بصورت دینامیک به PC هایی که به آن وصل می‌شوند، اختصاص می‌دهد. اگر یک ISP دارای n آدرس IP باشد، می‌تواند در هر لحظه (حداکثر) n کاربر متصل به اینترنت داشته باشد (البته تعداد کل مشترکان آن می‌تواند خیلی بیشتر باشد). یکی از بسته‌های NCP که درخواست IP می‌کند، این آدرس را به PC اختصاص می‌دهد.

از این لحظه به بعد PC ما درست مثل یک میزبان معمولی اینترنت است، و می‌تواند بسته‌های IP رد و بدل کند. وقتی کاربر ارتباط را قطع کند، NCP نیز ارتباط لایهٔ شبکه را قطع کرده و آدرس IP را آزاد می‌کند. پس از آن LCP ارتباط لایهٔ پیوند داده را قطع می‌کند، و کامپیوتر نیز به مودم دستور می‌دهد که گوشی را بگذارد و به ارتباط فیزیکی پایان دهد.

فرمت فریمهای PPP بسیار شبیه HDLC است (همیشه که نباید چرخ را از نو اختراع کرد). تفاوت اصلی PPP و HDLC اینست که، PPP **کاراکتر-گرا** (character-oriented) است نه بیت-گرا. بویژه، PPP از تکنیک **لاگذاری بایت** (byte stuffing) استفاده می‌کند، بنابراین تعداد بایتهای یک فریم همیشه عددی صحیح است. در PPP (برخلاف HDLC) نمی‌توان فریمی فرستاد که مثلاً 30.25 بایت داشته باشد. البته PPP می‌تواند (علاوه بر خطوط تلفن معمولی) روی SONET یا خطوط بیت-گرای HDLC نیز کار کند. فرمت فریم PPP را در شکل ۳-۲۷ ملاحظه می‌کنید.



شکل ۳-۲۷. فرمت فریم کامل PPP برای حالت بدون‌شماره.

تمام فریمهای PPP با بایت پرچم استاندارد HDLC (یعنی 01111110)، شروع می‌شوند، که اگر این بایت در داخل داده‌ها وجود داشته باشد از تکنیک لاگذاری بایت برای متمایز کردن آن استفاده می‌شود. بعد از آن فیلد *Address* می‌آید، که همیشه 11111111 است و مشخص می‌کند که تمام گیرنده‌ها باید این فریم را قبول کنند. با استفاده از این آدرس مشکل تخصیص آدرس‌های لینک داده نیز هم حل می‌شود.

بدنبال آدرس فیلد *Control* قرار دارد، که مقدار پیش‌فرض آن 00000011 است، و نشان می‌دهد که فریمها **بدون‌شماره** (unnumbered) هستند. بعبارت دیگر، در PPP فریمها شمارهٔ ترتیبی ندارند، و از فریم تصدیق دریافت نیز خبری نیست. البته در محیط‌های پُرنویز (مانند لینکهای بیسیم) می‌توان از فریمهای شماره‌دار استفاده کرد. جزئیات دقیق این حالت در استاندارد RFC 1663 مشخص شده، ولی در عمل بندرت از آن استفاده می‌شود.

از آنجائیکه در پیکربندی پیش‌فرض فیلدهای *Address* و *Control* همواره ثابت هستند، LCP مکانیزم خاصی را بین دو سر خط پیاده می‌کند که این دو بایت بکلی حذف و فریمها کوتاهتر شوند.

فیلد چهارم PPP فیلد *Protocol* است، و مشخص می‌کند که دادهٔ موجود در فریم (قسمت Payload) از چه

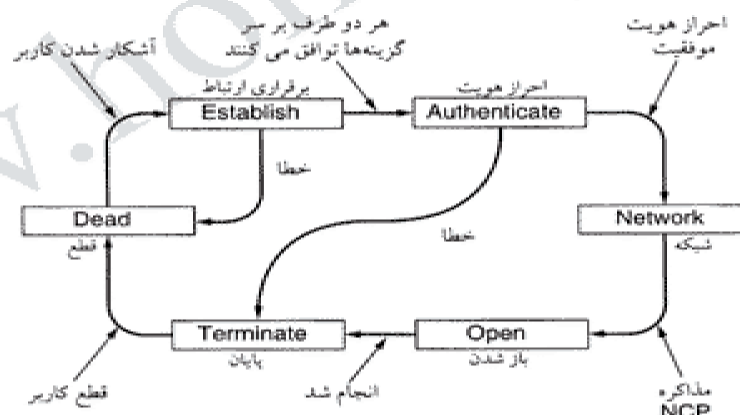
نوعیست. برای پروتکلهای مختلف از قبیل LCP ، NCP ، IP ، IPX ، Apple Talk و پروتکلهای دیگر کُدهایی تعریف شده است. پروتکلهایی که با 0 شروع می‌شوند، پروتکلهای لایهٔ شبکه (مانند IP ، IPX ، OSI CLNP و XNS) هستند؛ آنهایی که با 1 شروع می‌شوند، برای مذاکره در بارهٔ پروتکلهای دیگر بکار می‌روند (از جمله LCP ، و یک NCP خاص برای هر یک از انواع لایه‌های شبکه). اندازهٔ پیش‌فرض این فیلد 2 بایت است، ولی دو طرف می‌توانند از طریق LCP مذاکره کرده و اندازهٔ آنرا به 1 بایت تقلیل دهند.

طول فیلد *Payload* متغیر است، و حداکثر مقدار آن در مذاکرهٔ اولیه مشخص می‌شود. اگر دو طرف در مذاکرهٔ اولیه (هنگام برقراری ارتباط) بر سر این عدد به توافق نرسند، از عدد پیش‌فرض 1500 بایت استفاده خواهد شد. اگر مقدار دادهٔ ارسالی به این حد نرسد، لایهٔ پیوند داده بقیه را با کاراکتر خاصی پُر خواهد کرد.

بدنبال *Payload* فیلد *Checksum* می‌آید، که مقدار آن معمولاً 2 بایت است، ولی دو طرف می‌توانند بر سر جمع تطبیقی 2 بایتی هم توافق کنند.

بطور خلاصه، PPP یک مکانیزم فریم‌بندی چندپروتکلی است، که می‌توان از آن روی مودم، خطوط بیت-گرای HDLC ، SONET و سایر لایه‌های فیزیکی استفاده کرد. این پروتکل از کشف خطا، مذاکره برای گزینه‌های مطلوب، **فشرده‌سازی سرآیند** (header compression) (و در صورت نیاز، از ارتباط قابل‌اعتماد با فرمت HDLC) پشتیبانی می‌کند.

اکنون اجازه دهید ببینیم در PPP برقراری خط و قطع ارتباط چگونه انجام می‌شود. در شکل ۳-۲۸ مراحل (سادهٔ شدهٔ) برقراری خط، بکارگیری، و قطع آن نشان داده شده است. این مراحل برای ارتباط مودمی و یا مسیریاب-به-مسیریاب هر دو صادق است.



شکل ۳-۲۸. مراحل ساده شدهٔ برقرای و قطع خط در پروتکل PPP .

در شروع کار پروتکل خط در وضعیت *DEAD* است، و این به معنای آنست که هیچگونه کاریر یا ارتباطی در لایهٔ فیزیکی وجود ندارد. بعد از برقرای ارتباط در لایهٔ فیزیکی، خط به وضعیت *ESTABLISH* می‌رود. در این‌لحظه مذاکره بر سر گزینه‌های LCP شروع می‌شود، که اگر موفقیت‌آمیز باشد، به وضعیت *AUTHENTICATE* منجر می‌شود. حال دو طرف می‌توانند در صورت تمایل هویت یکدیگر را چک کنند. بعد از ورود به مرحلهٔ *NETWORK* ، لایهٔ شبکه با اجرای پروتکل NCP مناسب پیکربندی می‌شود. اگر این پیکربندی موفقیت‌آمیز

باشد، مرحلهٔ *OPEN* فرا رسیده و تبادل اطلاعات می‌تواند شروع شود. وقتی تبادل اطلاعات انجام شد، خط به مرحلهٔ *TERMINATE* رفته، و از آنجا دوباره به وضعیت *DEAD* برمی‌گردد و کاربر قطع می‌شود.

در مرحلهٔ *ESTABLISH*، مذاکره بر سر گزینه‌های پروتکل لینک داده توسط LCP انجام می‌شود. البته خود LCP هیچ علاقه‌ای به این گزینه‌ها ندارد، و فقط مکانیزمی برای مذاکره فراهم می‌آورد (بعبارت دیگر، پیشنهادی را فرستاده و بعد از دریافت پاسخ آنرا پذیرفته یا رد می‌کند). بررسی کیفیت خط (و اینکه آیا برای برقراری ارتباط باندازهٔ کافی خوب هست یا نه) نیز بر عهدهٔ LCP است. قطع خط (بعد از پایان یافتن کار) یکی دیگر از وظایف پروتکل LCP است.

در RFC 1661 یازده نوع فریم LCP تعریف شده است، که آنها را در شکل ۳-۲۹ ملاحظه می‌کنید. چهار فریم *Configure-* به **آغازکننده** (I) اجازه می‌دهند تا گزینه‌ای را پیشنهاد کند، و **پاسخ‌دهنده** (R) می‌تواند آنها را پذیرفته یا رد کند. اگر پاسخ‌دهنده گزینه‌ای را رد کند، می‌تواند پیشنهاد موردنظر خود را ارائه کرده، و یا اعلام کند که اساساً مایل نیست راجع به آن مذاکره کند. گزینه‌ها و پاسخ آنها جزئی از فریمهای LCP هستند.

توضیح	جهت	نام
لیست گزینه ها و مقادیر پیشنهادی	I → R	Configure-request
تمام گزینه ها قبول می شوند	I ← R	Configure-ack
بعضی از گزینه ها قبول نمی شوند	I ← R	Configure-nak
بعضی از گزینه ها مذاکره نمی شوند	I ← R	Configure-reject
تقاضای قطع خط	I → R	Terminate-request
قبول، خط قطع شد	I ← R	Terminate-ack
دریافت درخواست نامعلوم	I ← R	Code-reject
درخواست پروتکل نامعلوم	I ← R	Protocol-reject
لطفا این فریم را پس بفرست	I → R	Echo-request
فریم پس فرستاده شد	I ← R	Echo-reply
این فریم را نادیده بگیر (برای تست بود)	I → R	Discard-request

شکل ۳-۲۹. انواع فریمهای LCP.

کُدهای *Terminate-* برای قطع کردن خط (وقتی که دیگر نیازی به آن نیست) هستند. کُدهای *Code-reject* و *Protocol-reject* مشخص می‌کنند که پاسخ‌دهنده چیزی را دریافت کرده که معنی آنرا نمی‌فهمد. یکی از دلایل این وضعیت می‌تواند رخ دادن خطاهای کشف نشده روی خط باشد، ولی باحتمال بیشتر دلیل آن یکی نبودن ویرایش پروتکل LCP در دو سمت خط است. فریمهای *Echo-* برای تست کیفیت خط بکار برده می‌شوند. و بالاخره، فریم *Discard-request* برای دیباگ کردن بکار می‌آید (و نویسندهٔ پروتکل می‌تواند از آن برای تست پروتکل خود استفاده کند). گیرنده این قبیل فریمها را نادیده می‌گیرد، و هیچ عکس‌العملی به آنها نشان نمی‌دهد.

برخی از مهمترین گزینه‌هایی که می‌توان دربارهٔ آنها مذاکره کرد، عبارتند از: حداکثر اندازهٔ قسمت Payload فریم، فعال کردن احراز هویت و انتخاب پروتکل آن، فعال کردن مانیتورینگ کیفیت خط در طول عملیات، و انتخاب گزینه‌های فشرده‌سازی سرآیند.

دربارهٔ پروتکلهای NCP حرف کلی چندانی نمی‌توان زد. هر پروتکل NCP خاص یک نوع لایهٔ شبکه است، و بر سر گزینه‌های پیکربندی آن مذاکره می‌کند. برای مثال، در یک لایهٔ شبکهٔ IP احتمالاً مهمترین گزینه تخصیص آدرس IP است.

۷ـ۳　خلاصه

وظیفهٔ لایهٔ پیوند داده تبدیل استریم خام بیت‌های لایهٔ فیزیکی به استریمی از فریمها، و هدایت این استریم به لایهٔ شبکه است. روشهای مختلفی برای فریم‌بندی وجود دارد، که شمارش کاراکتر، لاگذاری بایت، و لاگذاری بیت از آن نمونه است. پروتکلهای لینک داده می‌توانند خطاها را کنترل کرده، و در صورت نیاز فریمهای معیوب را مجدداً ارسال کنند. برای جلوگیری از غرق شدن گیرنده‌های کُند در سیلاب داده‌های فرستنده‌های پُرسرعت، لایهٔ پیوند داده جریان داده‌ها را نیز کنترل می‌کند. یکی از پروتکلهایی که کنترل خطا و کنترل جریان در آن لحاظ شده، پروتکل پنجرهٔ لغزنده است.

پروتکلهای پنجرهٔ لغزنده را می‌توان بر حسب اندازهٔ پنجرهٔ ارسال و دریافت دسته‌بندی کرد. وقتی اندازهٔ این پنجره‌ها هر دو 1 باشد، پروتکل توقف-انتظار نام دارد. اگر پنجرهٔ ارسال و دریافت (برای اجتناب از قفل شدن فرستنده در محیطهای با تأخیر زیاد) بزرگتر از 1 باشند، گیرنده می‌تواند فریمهای خارج از نظم را بکلی دور انداخته و یا آنها را بافر کند.

در این فصل چندین پروتکل را مورد بررسی قرار دادیم. پروتکل 1 برای محیطهای عاری از خطا طراحی شده، و می‌تواند انتقال اطلاعات را با هر سرعتی انجام دهد. در پروتکل 2 محیط همچنان بدون خطا فرض شده، ولی قادر است جریان داده‌ها را کنترل کند. در پروتکل 3 برای کنترل خطا از شمارهٔ ترتیبی فریمها و الگوریتم توقف-انتظار استفاده شده است. ترافیک در پروتکل 4 می‌تواند دوطرفه باشد، و در آن تکنیکی بنام سواری مجانی معرفی شده است. پروتکل 5 از تکنیک پنجرهٔ لغزندهٔ «N تا به عقب برگرد» استفاده می‌کند. و بالاخره، در پروتکل 6 از تکنیکهای تکرار انتخابی و تصدیق دریافت منفی استفاده کردیم.

برای تست صحت و کارایی پروتکلها، روشهای مختلف مدلسازی (از جمله مدل ماشین حالت محدود و مدل شبکهٔ پتری) را معرفی کردیم.

شبکه‌های بسیاری در لایهٔ پیوند داده از پروتکلهای بیت-گرا استفاده می‌کنند، که از آن میان می‌توان به SDLC ، HDLC ، ADCCP ، یا LAPB اشاره کرد. در تمام این پروتکلها ابتدا و انتهای فریم با یک بایت پرچم مشخص می‌شود، و اگر این بایت در داده‌ها موجود باشد، از تکنیک لاگذاری بیت برای مشخص کردن آن استفاده می‌شود. همهٔ این پروتکلها از یک پنجرهٔ لغزنده برای کنترل جریان سود می‌برند. در اینترنت نیز PPP بعنوان پروتکل اصلی لایهٔ پیوند داده در خطوط نقطه-به-نقطه کاربرد گسترده‌ای دارد.

مسائل

۱. یک بسته از لایهٔ بالاتر به ۱۰ فریم تقسیم شده، و احتمال اینکه هر یک از آنها سالم به مقصد برسد، ۸۰٪ است. اگر در لایهٔ پیوند داده کنترل خطا انجام نشود، این پیام چند بار باید فرستاده شود تا تمام آن صحیح و سالم به مقصد برسد؟

۲. در لایهٔ پیوند داده از کُدگذاری زیر استفاده شده است:

A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000

در هر یک از حالتهای زیر، توالی بیت (باینری) فریم چهار کاراکتری A B ESC FLAG را نشان دهید:

(الف) شمارش کاراکتر.

(ب) بایت پرچم با لاگذاری بایت.

(ج) بایت پرچم در ابتدا و انتها، با لاگذاری بیت.

۳. با تکنیکی که در کتاب شرح داده شد، قطعه دادهٔ A B ESC C ESC FLAG FLAG D در وسط یک استریم ظاهر شده است. خروجی لاگذاری این قطعه چیست؟

۴. یکی از همکلاسی‌های شما عقیده دارد که قرار دادن یک بایت پرچم در ابتدای فریم و یکی در انتهای آن کار بیهوده‌ایست، و یک بایت کاملاً کفایت می‌کند (این بایت انتهای یک فریم، و ابتدای فریم بعدی محسوب خواهد شد). نظر شما چیست؟

۵. می‌خواهیم رشتهٔ 0111101111101111110 را از طریق لایهٔ پیوند داده ارسال کنیم. این رشته بعد از لاگذاری بیت به چه شکلی در می‌آید؟

۶. آیا در هنگام استفاده از لاگذاری بیت احتمال دارد که خطایی (ازقبیل اضافه، کم شدن و یا تغییر یک بیت) رخ دهد، ولی جمع تطبیقی آنرا کشف نکند؟ اگر خیر، چرا؟ اگر بلی، چگونه؟ آیا طول جمع تطبیقی در اینجا نقشی دارد؟

۷. آیا فکر می‌کنید حالتی وجود دارد که یک پروتکل حلقهٔ-باز (مانند کُد همینگ) بر پروتکلهای مبتنی بر بازخور (که در این فصل دیدید) ارجحیت داشته باشد؟

۸. برای اطمینان بیشتر، بجای یک بیت توازن از کُدی استفاده می‌کنیم که یک بیت توازن برای بیت‌های فرد و یک بیت توازن دیگر برای بیت‌های زوج دارد. فاصلهٔ همینگ این کُد چقدر است؟

۹. برای ارسال یک پیام ۱۶ بیتی از کُد همینگ استفاده کرده‌ایم. برای آنکه گیرنده بتواند تمام خطاهای تک‌بیتی را کشف و تصحیح کند، به چند بیت چک‌کننده نیاز داریم. روش کار را برای پیام 1101001100110101 نشان دهید. فرض کنید در این کُد همینگ از توازن زوج استفاده شده است.

۱۰. می‌خواهیم یک بایت ۸ بیتی با مقدار باینری 10101111 را با استفاده از روش همینگ توازن زوج کُد کنیم. خروجی چیست؟

۱۱. یک کُد همینگ ۱۲ بیتی با مقدار هگزادسیمال 0xE4F به گیرنده می‌رسد. مقدار هگزادسیمال اولیه چه بوده است؟ فرض کنید بیش از یک بیت خطا رخ نداده است.

۱۲. یکی از تکنیکهای تشخیص خطا، ارسال داده‌ها بصورت ماتریسی از n سطر و k ستون است که هر سطر و ستون دارای بیت‌های توازن خاص خود است. آخرین بیت در منتهی‌الیه سمت راست بیتی است که سطر و ستون مربوطه را چک می‌کند. آیا این روش می‌تواند خطاهای تک‌بیتی را کشف کند؟ خطاهای دوبیتی را چطور؟ خطاهای سه‌بیتی را چطور؟

۱۳. ماتریسی با n سطر و k ستون دارای بیت‌های توازن افقی و عمودی است. فرض کنید در هنگام انتقال اطلاعات ۴ بیت تغییر کرده است. احتمال کشف نشدن این خطا را بصورت یک عبارت ریاضی استخراج کنید.

۱۴. حاصل تقسیم $x^7 + x^5 + 1$ بر چندجمله‌ای مولد $x^3 + 1$ چیست؟

۱۵. می‌خواهیم با استفاده از تکنیک CRC استریم 10011101 را ارسال کنیم. چندجمله‌ای مولد $x^3 + 1$ است. استریم فرستاده شده چیست؟ فرض کنید بیت سوم از سمت چپ در حین ارسال معکوس می‌شود. نشان دهید که گیرنده می‌تواند این خطا را کشف کند.

۱۶. پروتکلهای لینک داده همیشه CRC را بجای سرآیند در پی‌آیند پیام قرار می‌دهند. چرا؟

۱۷. یک کانال 4-kbps دارای تأخیر انتشار 20 msec است. تا چه اندازهٔ فریمی کارایی پروتکل توقف-انتظار بیش از ٪۵۰ است؟

۱۸. یک ترانک T1 بطول 3000 km از فریمهای 64 بایتی و پروتکل 5 استفاده می‌کند. اگر تأخیر انتشار 6μsec/km باشد، تعداد بیتهای شمارهٔ ترتیبی چقدر باید باشد؟

۱۹. آیا در پروتکل 3 فرستنده می‌تواند تایمری را که در حال کار است، از نو شروع کند؟ اگر بلی، در چه موقعیتی؟ اگر خیر، چرا؟

۲۰. فرض کنید در یک پروتکل پنجرهٔ لغزنده تعداد بیتهای شمارهٔ ترتیبی آنقدر زیاد است که «برگشت»

هرگز رخ نمی‌دهد. چه رابطه‌ای باید بین لبه‌های پنجره‌ها و اندازهٔ پنجره (که ثابت، و در فرستنده و گیرنده یکی است) برقرار باشد؟

۲۱. اگر روال *between* در پروتکل 5 بجای $a \le b < c$ شرط $a \le b \le c$ را چک کند، چه تأثیری روی درستی یا کارایی پروتکل خواهد گذاشت؟ توضیح دهید.

۲۲. در پروتکل 6 وقتی یک فریم به گیرنده می‌رسد، چک می‌کند که آیا شمارهٔ ترتیبی آن همانی است که باید باشد، و آیا *no_nak* مقدار true دارد یا خیر. اگر هر دو شرط true باشند، یک NAK فرستاده می‌شود؛ در غیر اینصورت، تایمر کمکی راه‌اندازی می‌شود. اگر قسمت else را حذف کنیم، چه تأثیری روی درستی پروتکل می‌گذارد؟

۲۳. فرض کنید حلقهٔ while سه دستوری نزدیک به انتهای پروتکل 6 را حذف کرده‌ایم. آیا این کار بر درستی پروتکل اثر می‌گذارد، یا فقط کارایی آنرا تحت تأثیر قرار می‌دهد؟ توضیح دهید.

۲۴. فرض کنید بخش case مربوط به خطاهای جمع تطبیقی را از دستور switch پروتکل 6 حذف کرده‌ایم. این کار چه تأثیری بر عملکرد پروتکل خواهد گذاشت؟

۲۵. کُد *frame_arrival* در پروتکل 6 بخشی برای NAK ها دارد. این بخش زمانی اجرا می‌شود که فریم ورودی یک NAK باشد و شرط دیگری وجود داشته باشد. سناریویی را شرح دهید که در آن وجود این شرط دوم الزامی باشد.

۲۶. فرض کنید می‌خواهید برای لایهٔ پیوند داده خطی برنامه بنویسید که در آن داده‌ها فقط به سمت شما می‌آیند، و شما هیچ چیزی نمی‌فرستید. سمت مقابل از پروتکل HDLC با شمارهٔ ترتیبی ۳ بیتی، و پنجرهٔ ۷ فریمی استفاده می‌کند. برای بالا بردن کارایی سیستم، تصمیم گرفته‌اید حداکثر فریمهای خارج از نظم ممکنه را بافر کنید، ولی مجاز به دستکاری در نرم‌افزار سمت فرستنده نیستید. آیا می‌توان پنجرهٔ دریافتی بزرگتر از 1 داشت، و تضمین کرد که این پروتکل هرگز با شکست مواجه نشود؟ اگر بلی، بزرگترین پنجره‌ای که می‌توان با اطمینان بکار برد، چقدر است؟

۲۷. پروتکل 6 را روی یک خط 1-Mbps عاری از خطا در نظر بگیرید. حداکثر اندازهٔ فریم 1000 بیت است. بسته‌ها با فواصل یک ثانیه‌ای تولید می‌شوند. فاصلهٔ زمانی انقضای تایمر 10 msec است. اگر تایمر مخصوص تصدیق دریافت را حذف کنیم، انقضاهای غیرلازم رخ خواهد داد. یک پیام با طول متوسط چند بار باید مجدداً ارسال شود؟

۲۸. در پروتکل 6 داریم: $MAX_SEQ = 2^n - 1$. با اینکه این شرط برای استفادهٔ بهینه از بیت‌های سرآیند آشکارا مناسب است، نشان نداده‌یم که الزامی هم هست. اگر، برای مثال $MAX_SEQ = 4$ ، آیا باز هم این پروتکل بدرستی کار می‌کند؟

۲۹. با استفاده از یک کانال ماهواره‌ای 1-Mbps که زمان رسیدن سیگنال از زمین به ماهواره 270 msec است، فریمهای 1000 بیتی ارسال می‌کنیم. فریمهای تصدیق دریافت همیشه با سواری مجانی برمی‌گردند، و سرآیند فریمها بسیار کوتاه است. حداکثر نرخ مصرف قابل دستیابی در پروتکلهای زیر چقدر است؟

(الف) پروتکل توقف-انتظار.

(ب) پروتکل 5 .

(ج) پروتکل 6 .

۳۰. مقدار اتلاف پهنای باند پروتکل 6 را (مربوط به سرآیند و ارسال مجدد) در یک کانال ماهواره‌ای 50-kbps پُرترافیک، با فریمهایی متشکل از 40 بیت سرآیند و 3960 بیت داده، محاسبه کنید. فرض کنید زمان رسیدن سیگنال از زمین به ماهواره 270 msec است؛ فریمهای ACK هرگز ارسال نمی‌شوند؛

فریمهای NAK دارای طولی معادل 40 بیت هستند؛ نرخ خطا در فریمهای داده ۱٪، و در فریمهای NAK قابل چشم‌پوشی است؛ و شماره‌های ترتیبی 8 بیتی هستند.

۳۱. می‌خواهیم روی یک کانال ماهواره‌ای عاری از خطا با ظرفیت 64 kbps فریمهای 512 بایتی (در یک جهت)، با فریمهای تصدیق دریافت بسیار کوتاه (در جهت دیگر)، بفرستیم. اگر اندازهٔ پنجره 1، 7، 15 و 127 باشد، حداکثر ظرفیت خط چقدر خواهد بود؟ زمان ارسال سیگنال از زمین به ماهواره را 270 msec بگیرید.

۳۲. یک کابل T1 بطول 100 km را در نظر بگیرید. سرعت انتشار امواج در این کابل 2/3 سرعت نور در خلاء است. چند بیت در این کابل جا می‌شود؟

۳۳. فرض کنید پروتکل ۴ را با ماشین حالت محدود مدل کرده‌ایم. هر ماشین چند حالت دارد؟ کانال مخابراتی چند حالت دارد؟ سیستم کامل (دو ماشین و یک کانال مخابراتی) چند حالت دارد؟ از خطاهای جمع تطبیقی صرفنظر کنید.

۳۴. توالی آتش شبکهٔ پتری شکل ۳-۲۳ که با توالی حالت (000), (01A), (01-), (010), (01A) در شکل ۳-۲۱ متناظر است، را در نظر بگیرید. توضیح دهید که این توالی چه چیزی را نشان می‌دهد.

۳۵. شبکهٔ پتری دستورات گذار $AC \rightarrow B$، $AC \rightarrow B$، $CD \rightarrow E$، و $E \rightarrow CD$ را رسم کنید. از این شبکهٔ پتری گراف دسترسی حالت محدود برای رسیدن به حالت ACD را رسم کنید. این گراف چه مفهوم شناخته‌شده‌ای را مدل می‌کند؟

۳۶. پروتکل PPP از HDLC مشتق شده، که در آن برای جلوگیری از تفسیر اشتباه بایت پرچم در داده‌ها از تکنیک لاگذاری بیت استفاده می‌شود. یک دلیل بیاورید که چرا PPP از لاگذاری بایت استفاده می‌کند، نه لاگذاری بیت.

۳۷. حداقل سربارهٔ یک بستهٔ IP که با PPP فرستاده شده، چقدر است؟ فقط سربارهٔ PPP را در نظر بگیرید، نه سربارهٔ سرآیند IP را.

۳۸. هدف از این تمرین آزمایشگاهی پیاده‌سازی یک مکانیزم کشف خطا با استفاده از الگوریتم CRC (که در متن کتاب توضیح دادیم) است. دو برنامه بنویسید: مولد (generator)، و تست‌کننده (verifier). برنامهٔ مولد یک رشتهٔ n بیتی از 0 ها و 1 ها را از ورودی استاندارد (با فرمت ASCII) می‌خواند. خط دوم یک چندجمله‌ای k بیتی است، که آن هم بصورت ASCII از ورودی خوانده می‌شود. برنامه خروجی خود را، که تشکیل شده از $n + k$ بیت 0 و 1 (و در واقع همان ورودی کُد شده است)، روی خروجی استاندارد می‌نویسد. (برنامهٔ مولد این خروجی را هم با فرمت ASCII بیرون می‌دهد.) برنامهٔ تست‌کننده خروجی مولد را گرفته، و با پیامی نشان می‌دهد که آیا این رشته درست است یا خیر. برای تست این دو برنامه، یک برنامهٔ کمکی دیگر (بنام alter) بنویسید که یکی از بیتهای خط اول را معکوس کند و سایر بیت‌ها را بهمان صورت باقی بگذارد (شمارهٔ بیتی که باید معکوس شود را - با آغاز شمارش از سمت چپ - بصورت آرگومان ورودی به این برنامه بدهید). با نوشتن

```
generator < file | verifier
```

برنامه باید پاسخ «درست است» بدهد، اما با دستور

```
generator < file | alter arg | verifier
```

باید پیام «درست نیست» بگیرید.

۳۹. برنامه‌ای بنویسید که رفتار یک شبکهٔ پتری را شبیه‌سازی کند. این برنامه باید دستورات گذار، و لیستی از حالتهای لایهٔ شبکه (هنگام ارسال و دریافت یک بستهٔ جدید) را بعنوان ورودی بخواند. برنامه باید از حالت اولیه (که آنرا هم از ورودی می‌خواند) یکی از گذارهای فعال را بصورت تصادفی آتش کند، و سپس چک کند که آیا هر یک از ماشینها ۲ بسته قبول می‌کند، بدون این که بین آنها یک بستهٔ جدید بفرستد.

۴ زیر لایه نظارت بر دسترسی به رسانه انتقال

به گونه‌ای که در فصل اوّل اشاره کردیم، شبکه‌ها را می‌توان به دو رده تقسیم‌بندی کرد: آنهائی که از اتصالات نقطه‌به‌نقطه استفاده می‌کنند و آنهائی که از کانالهای پخش فراگیر (Broadcast) بهره می‌گیرند. در این فصل به شبکه‌های پخش فراگیر و پروتکلهای آنها خواهیم پرداخت.

در هر شبکهٔ فراگیر مسئله اصلی آنست که وقتی برای دسترسی به کانال انتقال ، رقابت وجود دارد چگونه می‌توان تعیین کرد که چه کسی باید از کانال استفاده کند. برای روشن‌تر شدن قضیه یک کنفرانس تلفنی را با حضور شش نفر، در نظر بگیرید که از طریق شش خط تلفن بهم متصل شده و نشستی را ترتیب داده‌اند و هر کدام قادرند با دیگران گفت و شنود داشته باشند. در چنین وضعیتی کاملاً طبیعی است که وقتی یک نفر صحبت خود را قطع می‌کند دو یا چند نفر به طور همزمان شروع به حرف زدن نمایند و برای لحظاتی بی‌نظمی و اغتشاش بر جلسه حاکم شود. در ملاقاتهای رو در رو بکمک اشاره و علامت، از بروز بی‌نظمی احتراز می‌شود؛ مثلاً شخص مایل به گفتگو، دست خود را به علامت اجازه خواستن برای شروع سخن، بالا می‌برد. وقتی فقط یک کانال منفرد در اختیار است، تعیین نفر بعدی برای ارسال، دشوارتر خواهد بود. برای حلّ این مسئله پروتکلهای متعددی عرضه شده است که معرفی آنها شاکلهٔ این فصل را تشکیل می‌دهد. در ادبیات شبکه، کانالهای فراگیر گاهی با عناوین «**کانالهای با دسترسی چندگانه**» (Multiaccess Channel) یا «**کانالهای با دسترسی تصادفی**» (Random Access Channel) معرفی می‌شوند.

پروتکلهائی که برای تعیین نفر بعدی در استفاده از کانال مشترک کاربرد دارند متعلق به زیرلایه‌ای از لایهٔ پیوند داده‌ها هستند که اصطلاحاً زیرلایهٔ MAC (Medium Access Control) نامیده می‌شود. زیرلایهٔ MAC در شبکه‌های محلی که اغلب آنها از کانالهای مشترک به عنوان زیربنای ارتباط استفاده می‌کنند، از اهمیت ویژه‌ای برخوردار است. در مقابل، به غیر از شبکه‌های ماهواره‌ای ، تمام شبکه‌های WAN از خطوط نقطه‌به‌نقطه بهره گرفته‌اند [و در آنها زیرلایهٔ MAC جایگاهی ندارد]. از آنجایی که کانالهای با دسترسی چندگانه و شبکه‌های LAN کاملاً به یکدیگر مرتبط هستند لذا در این فصل بطور عامّ شبکه‌های LAN را بررسی می‌کنیم و موارد معدودی را نیز که مستقیماً مربوط به زیرلایهٔ MAC نمی‌شوند، بررسی خواهیم نمود.

از دیدگاه فنّی، زیرلایهٔ MAC بخش زیرین لایهٔ پیوندِ داده‌ها محسوب می‌شود و منطقاً می‌بایست قبل از آنکه در فصل سوم به پروتکل‌های نقطه‌به‌نقطه بپردازیم ابتدا این زیرلایه را بررسی می‌کردیم. ولیکن برای اغلب افراد ، فهم پروتکلهای دوطرفه ساده‌تر از پروتکل‌های چندطرفه است.[منظور از پروتکل دوطرفه پروتکلیست که در آن فقط و فقط دو ماشین درگیر مبادله داده هستند.] به همین دلیل در روند تشریح مطالب (که از لایه‌های پائین به

سمت بالا خواهد بود)، اندکی از این ترتیب تخطی کردیم.

۴ـ۱ مسئله تخصیص کانال

مضمون اصلی این فصل آنست که چگونه یک کانال مشترک و فراگیر را بین کاربران رقیب تقسیم نمائیم. در ابتدا به الگوهای تخصیص «ایستا» و «پویا» نگاهی می‌اندازیم؛ سپس به تشریح تعدادی از الگوریتمهای خاصّ در این زمینه خواهیم پرداخت.

۴ـ۱ـ۱ تخصیص ایستای کانال در شبکه‌های LAN و MAN

روش سنّتی در تخصیص یک کانال منفرد، (مثل خطوط اصلی تلفن - Telephone Trunk)، بین چندین کاربر که برای استفاده از آن رقابت می‌کنند، روش FDM (تسهیم در حوزهٔ فرکانس) است. اگر N کاربر حضور داشته باشند، پهنای باند کانال به N بخش متساوی تقسیم می‌شود و به هر کاربر یکی از این بخشها اختصاص داده می‌شود (شکل ۲ـ۳۱ را ببینید). از آنجایی که هر کاربر دارای یک باند فرکانس اختصاصی است لذا کاربران هیچگونه تداخل و مزاحمتی برای یکدیگر ندارند. وقتی تعداد کاربران ثابت و کم باشد و هر کدام نیز دارای بار سنگین (و بافر شدهٔ) ترافیک باشند (همانند مراکز سوئیچ تلفن)، روش FDM مکانیزمی ساده و کارآمد برای تخصیص کانال خواهد بود. ولیکن وقتی تعداد ارسال کنندگان زیاد و دائماً در حال تغییر باشد، یا ترافیک ارسالی آنها به صورت لحظه‌ای و انفجاری تولید شود، FDM مشکلات متعددی را بروز خواهد داد: اگر طیف فرکانسی کانال به N بخش مجزا تقسیم شود ولی تعداد کاربرانی که تمایل به ارسال و مخابرهٔ داده دارند کمتر از N باشد بخش ارزشمندی از این طیف فرکانسی تلف خواهد شد. اگر تعداد کاربران بیش از N باشد، برخی از آنها بدلیل کمبود پهنای باند، مجوز ارسال نخواهند داشت؛ حتی اگر برخی از کاربرانی که بدانها باند فرکانسی تخصیص داده شده به ندرت بخواهند چیزی ارسال یا دریافت کنند.

با این وجود، حتی اگر بتوان تعداد کاربران را عدد ثابت N فرض کرد، باز هم تقسیم ایستای طیف فرکانسی کانال منفرد به تعدادی زیرکانال، ذاتاً روشی ناکارآمد و بی‌کفایت تلقی می‌شود. مشکل اساسی آنست که وقتی برخی از کاربران، تقاضای ارسال نداشته باشند پهنای باند آنها هدر می‌رود. گذشته از آن در بسیاری از سیستم‌های کامپیوتری، ترافیک داده‌ها بشدّت انفجاری است (نسبت حداکثر ترافیک به متوسط ترافیک در حدود 1000:1 است). در نتیجه اکثر اوقات کانالها خالی و بلااستفاده باقی می‌مانند.

کارآئی بسیار ضعیف روش FDM را می‌توان با یک محاسبهٔ ساده در **«نظریهٔ صفّ»** (Queuing Theory) اثبات کرد. برای شروع، فرض کنید بخواهیم زمان متوسط تاخیر T را برای کانالی محاسبه کنیم که در آن نرخ ارسال C بیت بر ثانیه، نرخ دریافت فریمها λ فریم بر ثانیه و طول هر فریم تصادفی است و از تابع چگالی احتمال نمائی با میانگین $1/\mu$ بیت بر فریم تبعیت می‌کند.

با این پارامترها، نرخ دریافت فریمها λ فریم بر ثانیه و نرخ سرویس‌دهی $\mu.C$ فریم بر ثانیه خواهد بود. با استفاده از نظریهٔ صفّ، می‌توان نشان داد که اگر نرخ دریافت و زمان سرویس‌دهی به فریمها از تابع توزیع پوآسون تبعیت کند خواهیم داشت:

$$T = \frac{1}{\mu.C-\lambda}$$

بعنوان مثال اگر C معادل 100 مگابیت بر ثانیه، متوسط طول فریمها (یعنی $1/\mu$) معادل 10000 بیت و نرخ دریافت فریمها (یعنی λ) معادل 5000 فریم بر ثانیه باشد، متوسط تاخیر هر فریم $T = 200\mu s$ خواهد بود. دقت کنید که اگر از تاخیر صفّ (یعنی تاخیر انتظار فریم) صرف‌نظر می‌کردیم و زمان ارسال 10000 بیت را بر روی یک شبکهٔ

100Mbps محاسبه می‌نمودیم به پاسخ اشتباهِ 100 میکروثانیه می‌رسیدیم. این نتیجه فقط زمانی صدق می‌کند که هیچ رقابتی در بدست گرفتن کانال وجود نداشته باشد. حال بیائید این کانال منفرد را به N زیرکانال مستقل با ظرفیت C/N بیت بر ثانیه تقسیم کنیم. در این حالت نرخ متوسط ورودی به هر یک از این زیرکانالها λ/N خواهد بود. با محاسبه مجددِ T به دست می‌آوریم:

$$T_{FDM} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu.C-\lambda} = N.T$$ رابطهٔ (۴-۱)

متوسط تاخیر در FDM ، N برابر بیشتر از زمانی است که تمام فریمها به ترتیب در یک صفِ مرکزی طولانی و مرتب شده، پشت سرهم ارسال شوند.

تمام استدلالات و مباحثاتی که در مورد روش FDM اعمال کردیم دقیقاً در مورد TDM نیز صدق می‌کند. به هر کاربر بصورت ثابت یکی از N برش زمانی (slot time) اختصاص داده می‌شود. اگر یک کاربر از برش زمانیِ تخصیص یافته به خود استفاده نکند، آن زمان بلااستفاده مانده و هدر خواهد رفت. با استفادهٔ مجدد از مثال قبلی، اگر یک شبکهٔ 100Mbps را به ده شبکهٔ 10Mbps تقسیم کرده و هر کاربر از یکی از آنها استفاده کند، میانگین تاخیر از 200 میکروثانیه به 2 میلی‌ثانیه افزایش خواهد یافت.

از آنجاکه هیچیک از روشهای معمول و ایستای تخصیص کانال در محیطهای با ترافیک انفجاری کار نخواهد کرد لذا در ادامه به بررسی روشهای پویا می‌پردازیم.

۴-۱-۲ تخصیص پویای کانال در LAN و MAN

قبل از آنکه به اوّلین روش از روشهای متعدّدِ تخصیص کانال بپردازیم دسته‌بندی و فرموله کردن مسائل و مشکلات تخصیص کانال مفید خواهد بود. کل کاری که باید برای تخصیص کانال انجام شود مبتنی بر پنج فرض اساسی است که در زیر تشریح شده‌اند:

۱. ***مدل ایستگاه* (Station Model):** این مدل شامل N ایستگاهِ مستقل (مثل کامپیوتر، تلفن یا دستگاه‌های مخابرهٔ شخصی) است که در هر کدام از آنها یک برنامه یا کاربر، فریمهایی را برای ارسال تولید می‌کند. برخی از اوقات به ایستگاه‌ها، «پایانه» (یا ترمینال) نیز گفته می‌شود. احتمال آنکه در بازهٔ زمانی ΔT فریمی تولید شود، $\lambda.\Delta T$ است که در آن λ یک مقدار ثابت است (در حقیقت λ میانگین نرخ تولید فریمهای جدید است). بمحض آنکه فریمی تولید گردد، ایستگاه متوقف شده و تا زمانی که آن فریم به صورت موفقیت‌آمیز ارسال نشود کاری انجام نمی‌دهد.

۲. ***فرض کانال منفرد* (Single Channel Assumption):** در این حالت تنها یک کانال منفرد و مشترک برای مخابرهٔ داده در اختیار ایستگاه‌ها است. تمام ایستگاه‌ها می‌توانند اطلاعات خود را برروی این کانال بفرستند یا از آن دریافت کنند. از دیدگاه سخت‌افزاری تمام ایستگاه‌ها همارز و معادل یکدیگرند؛ اگرچه ممکن است در نرم‌افزارِ پروتکل به هر ایستگاه، اولویتی خاصّ داده شود.

۳. ***فرض تصادم* (Collision Assumption):** هرگاه دو فریم بطور همزمان [برروی کانال مشترک] ارسال شوند با یکدیگر تداخل کرده و سیگنال حاصل بی‌ارزش و نامعتبر خواهد بود. این رخداد اصطلاحاً **«تصادم»** (Collision) نامیده می‌شود. هر ایستگاه می‌تواند از وقوع تصادم آگاه گردد. فریمی که در حین ارسال آن تصادم رخ داده است باید از نو فرستاده شود. در این مدل هیچ خطائی به غیر از خرابی فریم در اثر تصادم لحاظ نمی‌شود.

۴. ***الف: مدل زمان پیوسته* (Continuous Time):** در این مدل، ارسال فریمها می‌تواند در هر لحظه از زمان

شروع شود و هیچگونه سیگنال ساعت مرکزی (سراسری) که زمان را به برشهای گسسته و مجزا تقسیم کند، وجود ندارد.

۴.ب: «مدل زمان گسسته» (Slotted time): در این مدل، زمان به برشهای گسسته و مستقلی تقسیم می‌شود و ارسال فریم همیشه باید در ابتدای یکی از این برشهای زمانی انجام گیرد. در هر برش زمان ممکن است صفر، یک، یا چند فریم ارسال شود که به ترتیب: صفر فریم به معنای بیکار و بلااستفاده ماندن آن برش زمانی، یک فریم به معنای ارسال موفق و چند فریم معادل تصادم خواهد بود.

۵.الف: شنود سیگنال حامل (Carrier Sense): در این مدل، ایستگاه‌ها قبل از شروع به ارسال فریم خود، قادرند تشخیص بدهند که آیا کانال مشغول است یا آزاد؟ اگر ایستگاهی احساس کند که کانال مشغولَ است هیچگاه سعی در استفاده از آن نخواهد کرد مگر آنکه مجدداً کانال بیکار شود.

۵.ب: عدم شنود سیگنال حامل (No Carrier Sense): در این مدل، ایستگاه‌ها قادر نیستند قبل از استفاده از کانال، آنرا بشنوند و سیگنال روی آنرا احساس کنند؛ لذا فقط پس از فرستادن فریم می‌توان تعیین کرد که آیا ارسال موفق بوده یا تصادم پدیده آمده است.

اندکی توضیح در خصوص برخی از فرضیات فوق مفید خواهد بود: اوّلین بند اذعان می‌دارد که ایستگاه‌ها مستقل هستند و فریمها را با نرخ ثابتی تولید می‌کنند. [بعبارتی نرخ میانگین تولید فریمها ثابت است. -م] همچنین در این بند تلویحاً فرض شده که هر ایستگاه تنها یک کاربر یا برنامهٔ فعال دارد و بدین ترتیب هرگاه یک ایستگاه، متوقف (بلوکه) شود هیچ فریم جدیدی تولید نخواهد شد. مدلهای پیچیده دیگری نیز وجود دارد که در آنها ایستگاه‌ها اجازه دارند به صورت چندبرنامه‌ای (Multiprogrammed) تولید فریمها را ادامه بدهند (حتی وقتی که ایستگاه در انتظار ارسال فریم قبلی بلوکه شده است)؛ ولیکن تحلیل چنین ایستگاه‌هائی بسیار دشوار است.

فرض **کانال منفرد**، هستهٔ اصلی این مدل است. ایستگاه‌ها بجز یک کانال واحد و مشترک راهی برای مبادلهٔ اطلاعات ندارند. ایستگاه‌ها نمی‌توانند همانند یک کلاس درس با بالا بردن دست خود، از معلم کلاس برای صحبت کردن اجازه بگیرند!

فرض **تصادم** نیز محوری است، اگر چه برخی از سیستمها (بویژه سیستمهای مبتنی بر «طیف گسترده» - Spread Spectrum) از این فرض مستثنی بوده و نتایج شگفت‌آوری نیز به همراه دارند؛ همچنین در شبکه‌های توکن‌رینگ (Token Ring) یک نشانهٔ خاصّ (توکن) ایستگاه به ایستگاه می‌چرخد و هر ایستگاه که آنرا در اختیار بگیرد اجازهٔ ارسال فریم خود را خواهد داشت ولیکن در بخشهای بعدی به کانالهای منفردی خواهیم پرداخت که برای استفاده از آنها رقابت وجود دارد و تصادم پدید می‌آید.

در خصوص زمان ارسال فریمها، دو فرض (۱) زمان پیوسته (۲) زمان گسسته قابل اعمال است. برخی از سیستمها از مدل اوّل پیروی می‌کنند و برخی دیگر از مدل دوّم؛ بنابراین ما هر دو مدل را تحلیل خواهیم کرد. در هر سیستم فقط یکی از این مدلها قابل اعمال است.

همچنین در یک شبکه ممکن است سیگنال روی کانال احساس شود (فرض ۵-الف) یا شنود سیگنال میسر نباشد (فرض ۵-ب). شبکه‌های محلی (LAN) عموماً قادر به احساس سیگنال روی کانال هستند ولیکن در شبکه‌های بی‌سیم این مدل قابل استفاده نخواهد بود چراکه برخی از ایستگاه‌ها در محدودهٔ شنود سیگنال ایستگاه‌های دیگر نیستند. ایستگاه‌هائی که به کانالهای سیمی متصل هستند در مدل «شنود سیگنال حامل» (Carrier Sense) قرار می‌گیرند و قادرند به محض کشف پدیدهٔ تصادم، ارسال فریم را خاتمه بدهند. کشف تصادم در شبکه‌های بی‌سیم به دلایل فنی‌مهندسی به ندرت قابل انجام است. دقت کنید که کلمهٔ **«حامل»**

(Carrier) در اینجا اشاره به یک سیگنال الکتریکی برروی کابل دارد.

۲ـ۴ پروتکلهای دسترسی چندگانه

الگوریتم‌های بیشماری در مورد تخصیص کانالهای با دسترسی چندگانه [کانالهای مشترک] معرفی شده‌اند. در بخشهای آتی نمونه‌هائی از جالبترین این پروتکلها را بررسی کرده و مثالهایی از کاربرد آنها ارائه خواهیم نمود.

ALOHA ۱ـ۲ـ۴

در اوان ۱۹۷۰، نورمن آبرامسون و همکاران او در دانشگاه هاوائی روشی جدید و جالب برای حل مسئله تخصیص و دسترسی به کانال‌های اشتراکی ابداع کردند. بعداً ، کار آنها توسط بسیاری از پژوهشگران ادامه یافت و تکمیل شد. (مرجع Abramson, 1985) اگر چه کار آقای آبرامسون که سیستم ALOHA نامیده شد براساس پخش امواج رادیوی زمینی بود ، ولیکن نظریه آنها در هر سیستمی که در آن کاربران برای استفاده از یک کانال مشترک ، به صورت ناهماهنگ رقابت می‌کنند، قابل اعمال و پیاده‌سازی است.

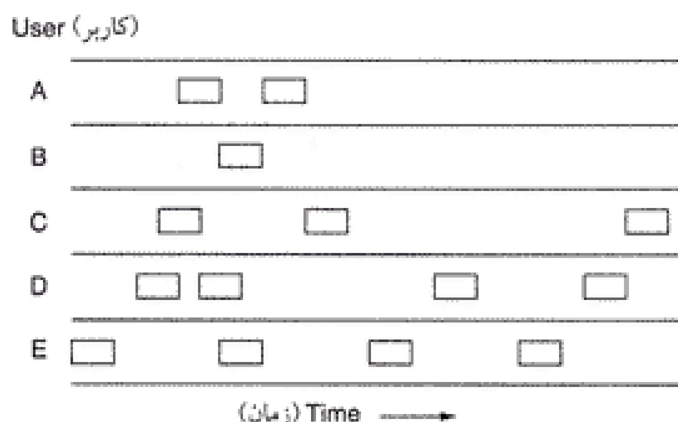
در اینجا دو نسخه متفاوت از ALOHA یعنی Pure ALOHA و Slotted ALOHA را تشریح خواهیم کرد. تفاوت این دو نسخه در تقسیم‌بندی زمان به برشهائی است که فریمها بتوانند در خلال یکی از آنها ارسال شوند. Pure ALOHA نیازی به هماهنگی زمانی (Time Synchronization) ندارد در حالیکه Slotted ALOHA نیازمند این هماهنگی است.

Pure ALOHA

ایده اصلی در سیستم ALOHA بسیار ساده است: کاربران اجازه دارند هر زمان که داده‌ای برای ارسال داشتند آنرا بفرستند. البته تصادمهایی رخ خواهد داد و فریمهایی که تصادم کنند از بین خواهند رفت. با این وجود در ALOHA ، یک «کانال بازگشت سیگنال» (Feedback) وجود دارد که فرستنده با گوش دادن به این کانال ، می‌تواند متوجه بروز تصادم و خرابی فریم شود. در شبکه‌های محلی LAN بروز تصادم به صورت سریع و آنی کشف می‌شود در حالیکه در شبکه‌های ماهواره‌ای یک ایستگاه پس از گذشت ۲۷۰ میلی ثانیه می‌تواند متوجه شود که آیا ارسال او موفق بوده یا نه! هرگاه به هر دلیلی امکان شنود سیگنال بازگشتی در حین ارسال وجود نداشته باشد بایستی دریافت فریمها توسط گیرنده تایید گردد. [با ارسال فریمهای مستقلی به نام Ack] هرگاه فریمی خراب شود، فرستندهٔ آن ، به اندازه یک زمان تصادفی صبر خواهد کرد و آن را مجدداً ارسال می‌کند. زمان انتظار قطعاً باید تصادفی باشد وگرنه تصادمها در یک دور نامتناهی تکرار خواهند شد. سیستمهایی که در آنها چندین کاربر از یک کانال مشترک به نحوی استفاده کنند که احتمال تصادم و تلاقی وجود دارد اصطلاحاً **«سیستم‌های رقابتی»** نامیده می‌شوند. در شکل ۴-۱ نمایشی از تولید فریم در سیستم ALOHA نشان داده شده است. در این شکل طول تمام فریمها را یکسان در نظر گرفته‌ایم زیرا اگر طول فریمها ، اندازهٔ ثابتی داشته باشند کارآئی و توان خروجی ALOHA حداکثر خواهد بود.

هرگاه دو فریم بطور همزمان برروی کانال ارسال شوند، تصادم رخ داده و هر دو خراب خواهند شد. حتی اگر اولین بیت از یک فریم جدید با آخرین بیت از فریم قبلی تداخل کند هر دو فریم بطور کامل خراب شده و بعداً باید از نو ارسال شوند زیرا کدهای کشف خطا نمی‌توانند (و نباید هم بتوانند) تشخیص بدهند خطا در کجا اتفاق افتاده و بدین ترتیب کلّ فریم بلااستفاده خواهد بود.

سوال جالبی که پیش می‌آید آنست که کارآئی کانال در ALOHA چقدر است؟ بعبارت دیگر می‌خواهیم بدانیم در این محیطِ نامنظم و هرج‌ومرج ، چند درصد از کلّ فریمهای ارسالی از تصادم جان سالم به در می‌برند؟



شکل ۴-۱. در Pure ALOHA فریمها در زمانهای کاملاً دلخواه ارسال می‌شوند.

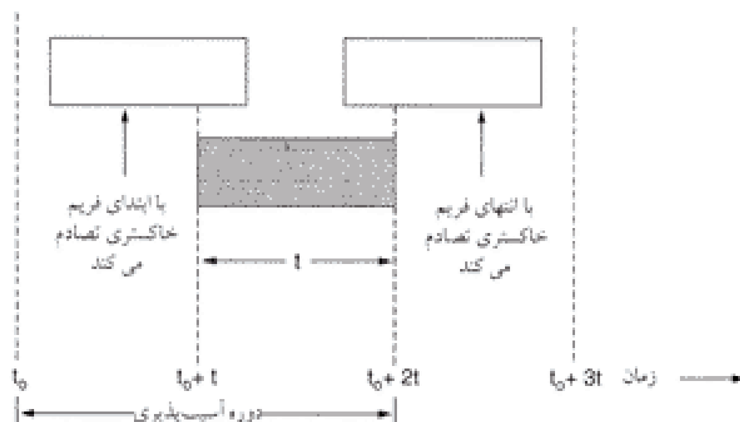
در ابتدا فرض می‌کنیم که تعدادی نامتناهی از کاربران در شبکه وجود دارند که پشت کامپیوترهای خود نشسته‌اند؛ هر کاربر در یکی از دو وضعیت «تایپ» یا «انتظار» قرار دارد. هرگاه تایپ یک خط به اتمام رسید و کلید Enter فشار داده شد، کاربر از تایپ دست می‌کشد و در انتظار پاسخ باقی می‌ماند. ایستگاه، فریم حاوی این خط را بلافاصله بر روی کانال می‌فرستد و بررسی می‌کند که آیا ارسال موفقیت‌آمیز بوده است؟ اگر فرآیند ارسال موفق باشد کاربر پاسخ خود را دریافت کرده و عمل تایپ را از سرمی‌گیرد و در غیر اینصورت، کاربر باز هم منتظر می‌ماند تا ارسال فریم آنقدر تکرار شود تا بالاخره یکی از آنها به سلامت به مقصد برسد.

«**زمان فریم**» (Frame Time)، مقدار زمانی است که طول می‌کشد تا یک فریم با طول ثابت و استاندارد ارسال شود. (به عبارت دیگر این زمان معادل با طول فریم تقسیم بر نرخ ارسال خواهد بود.) در اینجا فرض را بر آن می‌گذاریم که تعداد نامحدودی کاربر، مبتنی بر تابع توزیع پوآسون و با میانگین N فریم در واحد زمان، فریم‌های جدید تولید می‌کنند. (واحد زمان در اینجا زمان لازم برای ارسال یک فریم است). فرض بی‌نهایت بودن کاربران از آن جهت لازم است که مطمئن باشیم وقتی یک کاربر متوقف و منتظر می‌شود از تعداد کاربران کاسته نخواهد شد. اگر $N > 1$ باشد مجموع کاربران با نرخی بیشتر از ظرفیت کانال، فریم تولید کرده‌اند و تقریباً تمام فریمها در اثر تصادم نابود خواهند شد. برای آنکه کارآئی قابل ملاحظه‌ای داشته باشیم انتظار می‌رود که $0 < N < 1$ باشد. هر ایستگاه علاوه بر فریمهای جدید خود، فریمهایی را که قبلاً در اثر تصادم خراب شده‌اند، نیز ارسال می‌کند.

اجازه بدهید فرض کنیم که احتمال «تلاش برای ارسال k فریم در واحد زمان» نیز از تابع توزیع پوآسون با متوسط G تبعیت کند. (به خاطر داشته باشید که واحد زمان در اینجا زمان لازم جهت ارسال یک فریم -Frame Time- است.) بدیهی است که $G \geq N$. [زیرا N متوسط تولید فریم‌های جدید است در حالیکه G متوسط تولید فریم‌های جدید و فریم‌های قبلی خراب شده می‌باشد.-م]

در بار پائین (یعنی $N \approx 0$) تعداد تصادمها نیز اندک بوده و طبعاً ارسال مجدد فریمها ناچیز خواهد بود لذا $(G \approx N)$ است. در بار بالا تصادمهای زیادی رخ می‌دهد یعنی $G > N$ است؛ در هر شرایط بار، بازده مفید کانال (یعنی S) مساوی است با حاصل ضرب میزان بار (یعنی G) در احتمال موفقیت در ارسال (یعنی P_0) بنابراین داریم: $S = G \times P_0$ که در آن P_0 احتمال **عدم خرابی** یک فریم در اثر تصادم است.

فقط وقتی یک فریم در اثر تصادم خراب نخواهد شد که در زمان ارسال آن هیچ فریم دیگری ارسال نشود؛ به شکل ۴-۲ دقت کنید. تحت چه شرایطی فریمی که در شکل بصورت سایه‌دار نشان داده شده است سالم به مقصد خواهد رسید؟ t را زمان لازم برای ارسال یک فریم در نظر بگیرید. هرگاه کاربر دیگری در فاصلهٔ زمانی t_0 تا t_0+t فریمی را تولید و ارسال کرده باشد انتهای فریم او با ابتدای فریم سایه‌دار تصادم خواهد کرد. در حقیقت سرنوشت



شکل ۴-۲. دورهٔ آسیب‌پذیری برای فریم خاکستری.

فریم سایه‌دار به گذشته نیز بستگی دارد، حتی قبل از آنکه اولین بیت آن ارسال شود؛ چراکه در سیستم Pure ALOHA ، ایستگاه قبل از شروع به ارسال یک فریم قادر نیست به کانال گوش داده و متوجه شود که فریم دیگری در حال ارسال است. بدلیل مشابه اگر فریم جدیدی در فاصلهٔ زمان $t_0 + t$ تا $t_0 + 2t$ ارسال شود با انتهای فریم سایه‌دار در شکل تصادم خواهد کرد.

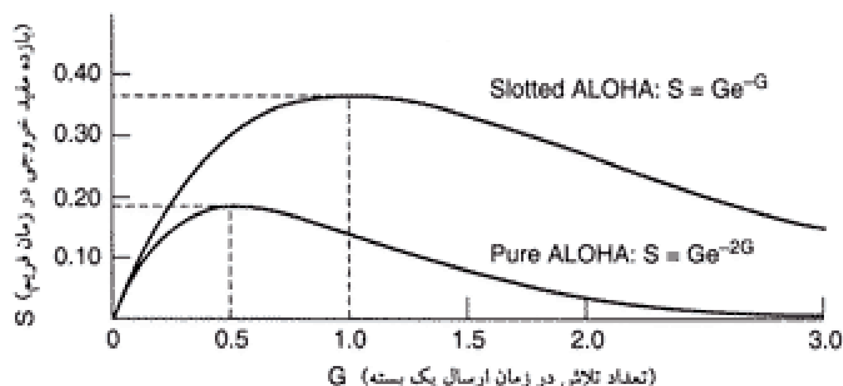
احتمال آنکه در زمان ارسال یک فریم [یعنی در زمان t] تعداد K فریم تولید شود از تابع توزیع پوآسون تبعیت می‌کند یعنی:

$$\Pr[k] = \frac{G^k \cdot e^{-G}}{k!} \qquad \text{رابطهٔ (۲-۴)}$$

بدین ترتیب احتمال ارسال صفر فریم معادل با e^{-G} است. در فاصلهٔ زمان ارسال دو فریم [یعنی 2t] میانگین فریم تولید شده معادل با 2G است. احتمال آنکه در طول «زمان آسیب‌پذیری» یک فریم، هیچ ترافیک دیگری تولید و ارسال نشود مساوی با $P_0 = e^{-2G}$ است. با اعمال رابطهٔ $S = G \times P_0$ به دست خواهیم آورد:

$$S = G \times e^{-2G}$$

در شکل ۴-۳، رابطهٔ بین ترافیک و بازده مفید کانال (Troughput) نشان داده شده است. بیشترین بازده به ازای $G = 0.5$ بدست می‌آید و در این حالت بازده کانال معادل $S = \frac{1}{2e}$، یعنی چیزی حدود 0.184 خواهد بود. به عبارت دیگر، بیشترین بهرهٔ مورد انتظار کانال (Channel Utilization) چیزی حدود ۱۸ درصد است. این مقدار بهرهٔ کانال چندان جالب نیست ولی در سیستمی که ایستگاه‌ها در زمانِ دلخواه فریم خود را ارسال می‌کنند نمی‌توان انتظار داشت بهرهٔ کانال صددرصد باشد. [یعنی هیچ تصادفی اتفاق نیفتد.]



شکل ۴-۳. بازده مفید کانال برحسب ترافیک عرضه شده در سیستم ALOHA.

Slotted ALOHA

در سال ۱۹۷۲ ، شخصی به نام روبرتز، روشی برای دو برابر کردن ظرفیت مفید سیستم ALOHA ارائه کرد.(مرجع Roberts, 1972) پیشنهاد وی مبنی بر آن بود که زمان به برشهای گسسته‌ای تقسیم شود و هر برش زمان معادل با زمان لازم برای ارسال یک فریم باشد. این روش مستلزم آن بود که کاربران محدودهٔ این برشهای زمانی (Time Slots) را به درستی بدانند. برای رسیدن به چنین هماهنگی می‌توان یک ایستگاه خاص را به خدمت گرفت تا در ابتدای هر برش زمانی سیگنالی همانند سیگنال ساعت منتشر نماید.

در روش رُوبرتز که امروزه به نام Slotted ALOHA مشهور شده است، برخلاف روش Pure ALOHA هیچ کامپیوتری مجاز نیست وقتی که کلید Enter (Carriage Return) فشار داده شد ، داده‌ها را بر روی کانال بفرستد؛ در عوض باید آنقدر منتظر بماند تا به آغاز برش زمان بعدی برسد. بنابراین روش پیوستهٔ Pure ALOHA به روش گسستهٔ Slotted ALOHA تبدیل شده است. از آنجایی که «دورهٔ آسیب‌پذیری» (یعنی زمانی که در خلال آن نباید فریم دیگری ارسال شود) نصف شده است لذا احتمال آنکه هیچ دادهٔ دیگری در خلال یک برش (اسلات) تولید نشود e^{-G} خواهد بود؛ بدین ترتیب بدست می‌آوریم:

$$S = G \times e^{-G} \qquad \text{رابطهٔ (۴-۳)}$$

به گونه‌ای که از شکل ۴-۳ مشهود است، در سیستم Slotted ALOHA ، بهرهٔ کانال در $G=1$ به مقدار حداکثر خود می‌رسد و در این نقطه بهرهٔ کانال معادل با $S=\frac{1}{e}$ (یعنی حدود 0.368) خواهد بود. اگر این سیستم در شرایط $G = 1$ عمل کند احتمال آنکه یک برش زمانی خالی باشد [و بتوان ارسال موفق داشت] حدود 0.368 خواهد بود. (طبق رابطهٔ ۴-۳) بیشترین موفقیتی که می‌توان از Slotted ALOHA انتظار داشت عبارتست از: ۳۷ درصد برای خالی ماندن یک اسلات، ۳۷ درصد برای ارسال موفق و ۲۶ درصد برای تصادم خواهد بود. اگر این سیستم با مقدارِ بیشترِ G کار کند، تعداد برشهای خالی کاهش یافته و میزان تصادمها به صورت نمائی افزایش خواهد داشت. [G: تلاش برای ارسال G عدد فریم در واحد زمان است و واحد زمان نیز زمان لازم برای ارسال یک فریم می‌باشد.] برای آنکه بررسی کنیم افزایش سریع تعداد تصادمها با افزایش G ، از کجا منشاء می‌گیرد ارسال یک فریم آزمایشی را مَد نظر قرار بدهید: احتمال آنکه این فریم از تصادم جان سالم به در ببرد (یعنی تمام ایستگاههای دیگر در این برش زمانی ساکت باشند) e^{-G} است و بالطبع احتمال تصادم معادل با $1 - e^{-G}$ خواهد بود. احتمال ارسال موفق فریم، منوط به k تلاش پیاپی خواهد بود (یعنی $k-1$ تصادم متوالی و نهایتاً یک ارسال موفق) یعنی:

$$P_K = e^{-G}.(1 - e^{-G})^{k-1}$$

پس از فشار داده شدن کلید Enter ، میانگین دفعات ارسال یعنی E ، معادل است با:

$$E = \sum_{k=1}^{\infty} k.P_k = \sum_{k=1}^{\infty} k.e^{-G}(1 - e^{-G})^{k-1} = e^{-G}$$

در نتیجه، از آنجایی که E به صورت نمائی با G ارتباط دارد اندکی افزایش در بار کانال ، بهرهٔ کانال را بشدت کاهش خواهد داد.

روش Slotted ALOHA بدلائلی که در بدو امر چندان مشهود و روشن به نظر نمی‌رسد، از اهمیت ویژه‌ای برخوردار است. این روش در دههٔ ۱۹۷۰ ابداع گردید، در چند سیستم آزمایشی و ابتدائی به کار گرفته شد و پس از آن تقریباً به دست فراموشی سپرده شد ولیکن وقتی دسترسی به اینترنت از طریق کابل اختراع شد ، ناگاه این مشکل بزرگ به میان آمد که چگونه می‌توان کانالی مشترک را به کاربران رقیب اختصاص داد. اینجا بود که Slotted ALOHA بار دیگر به صحنه آمد. پروتکلهایی بوده‌اند که اگرچه درست و موثر کار می‌کرده‌اند ولی بدلائل سیاسی [غیرعلمی] کنار گذاشته شده‌اند (مثلاً برخی از شرکتهای بزرگ علاقمندند که همه دنباله‌رُوی عملکرد آنها باشند)

ولیکن سالها بعد اشخاص زیرک و با هوش بدین حقیقت می‌رسند که این پروتکلهای فراموش شده می‌توانند مشکلات فعلی آنها را حلّ کنند. به همین دلیل در این فصل به پروتکلهای زیبا و جالبی خواهیم پرداخت که در حال حاضر کاربرد گسترده‌ای ندارند ولیکن ممکن است در کاربردهای آتی مفید واقع شوند؛ بشرط آنکه طراحان شبکه نسبت به آنها آگاهی داشته باشند. البته ما به بررسی پروتکل‌های متعددی نیز پرداخته‌ایم که در حال حاضر کاربرد بسیار گسترده‌ای دارند.

۴-۲-۲ پروتکلهای دسترسی چندگانه با قابلیت شنود سیگنال حامل (CSMA)

در روش Slotted ALOHA بیشترین بهرهٔ مفیدی که می‌توان بدست آورد 1/e [معادل 0.368] است. این بهره چندان جالب نیست چراکه در این روش هر ایستگاه بدون اعتنا به وضعیت بقیه ایستگاه‌ها و به دلخواه ارسال خود را انجام می‌دهد، لذا در این سیستم تعداد تصادمها زیاد خواهد بود. لیکن در شبکه‌های محلّی امکان آن وجود دارد که هر ایستگاه بتواند تشخیص بدهد دیگر ایستگاه‌ها چه می‌کنند و بر اساس این تشخیص عملکرد خود را تنظیم نماید. در چنین شبکه‌هایی می‌توان به بهرهٔ کانال بسیار بالاتر از 1/e دست یافت. در این بخش چندین پروتکل برای افزایش کارآئی و بهره کانال معرفی می‌کنیم.

پروتکلهایی که در آنها هر ایستگاه به سیگنال حامل روی کانال گوش داده و براساس وضعیت کانال عمل می‌کنند اصطلاحاً «**پروتکلهای شنود حامل**» (Carrier Sense Protocols) نامیده می‌شوند. تاکنون تعداد بی‌شماری از این پروتکلها معرفی شده‌اند. کلین‌راک و توباگی (۱۹۷۵) معدودی از این پروتکلها را تحلیل کرده‌اند. در ذیل چندگونه از پروتکلهای مبتنی بر شنود سیگنال حامل را بررسی خواهیم کرد.

Persistent and Nonpersistent CSMA

اوّلین پروتکل مبتنی بر شنود سیگنال حامل که در اینجا بررسی خواهیم کرد، روش 1-Persistent CSMA است. در این روش هرگاه یک ایستگاه، داده‌ای برای ارسال داشته باشد ابتدا به کانال گوش می‌دهد تا ببیند آیا در این لحظه کس دیگری در حال ارسال هست یا خیر. اگر کانال مشغول باشد ایستگاه آنقدر منتظر می‌ماند تا کانال آزاد شود؛ ولی اگر ایستگاه، کانال را آزاد تشخیص بدهد فریم خود را ارسال می‌کند. اگر تصادمی رخ بدهد ایستگاه به اندازهٔ یک زمان تصادفی صبر کرده و تمام مراحل را از نو آغاز می‌نماید. این پروتکل، اصطلاحاً 1-Persistent (پروتکل پافشاری بر ارسال) نامیده می‌شود چراکه وقتی یک ایستگاه کانال را آزاد تشخیص بدهد با احتمال ۱ ارسال خود را آغاز می‌کند. [یعنی اگر ایستگاه کانال را آزاد تشخیص بدهد یقیناً و به صورت غیرمشروط ارسال خود را آغاز می‌نماید. -م]

«**تاخیر انتشار**» (Propagation Delay) تاثیر بسزائی در کارآئی این پروتکل دارد. احتمال ناچیزی وجود دارد که دقیقاً پس از شروع ارسال فریم توسط یک ایستگاه، ایستگاه دیگری نیز آمادهٔ ارسال شده و کانال را بررسی و شنود نماید. اگر سیگنال ایستگاه اوّل هنوز به ایستگاه دوّم نرسیده باشد [بدلیل تاخیر انتشار]، دوّمی نیز کانال را آزاد تشخیص داده و ارسال خود را آغاز می‌کند و طبعاً منجر به تصادم (Collision) خواهد شد. هر چه تاخیر انتشار بیشتر باشد تاثیر مخرب آن بیشتر و منجر به کارآئی بدتر پروتکل خواهد شد.

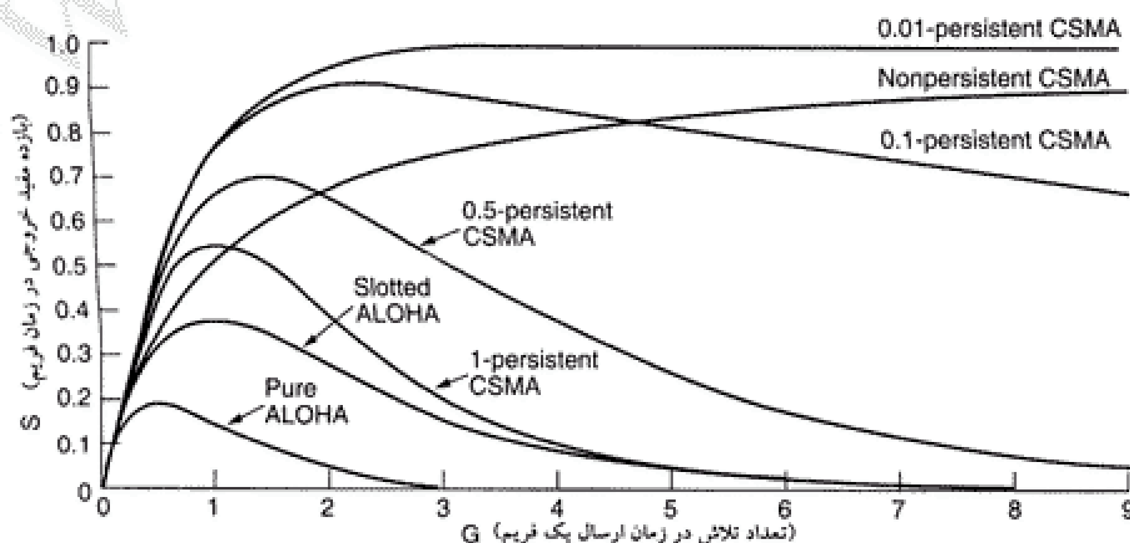
حتی اگر تاخیر انتشار صفر باشد باز هم «تصادم» وجود خواهد داشت: هرگاه دو ایستگاه در خلال ارسال ایستگاه ثالثی آمادهٔ ارسال فریم شوند هر دوی آنها مودبانه منتظر خاتمهٔ ارسال فریم جاری می‌شوند و به محض آزاد شدن کانال بطور همزمان ارسال فریم خود را آغاز می‌نمایند که منجر به تصادم خواهد شد. اگر این دو ایستگاه در ارسال فریم خود عجول و مصّر نبودند تصادم‌های کمتری رخ می‌داد! با این حال این پروتکل بسیار بهتر از Pure ALOHA عمل می‌کند زیرا در این پروتکل دو ایستگاه [با شنود کانال] از تلاقی با ارسال فریم ایستگاه در

حال ارسال اجتناب می‌کنند. می‌توان به صورت ذهنی کارآئی این روش را بسیار بیشتر از Pure ALOHA ارزیابی کرد. بدلیل مشابه، کارآئی این روش از Slotted ALOHA نیز بیشتر است.

دوّمین پروتکل که آن نیز مبتنی بر شنود سیگنال است Nonpersistent CSMA نام دارد. در این پروتکل، تلاش آگاهانه جهت ارسال برروی کانال با اصرار کمتری نسبت به پروتکل قبلی انجام می‌گیرد: هر ایستگاه قبل از ارسال، کانال را بررسی (شنود) می‌کند؛ اگر کسی دیگر در حال ارسال نباشد ایستگاه، فریم خودش را می‌فرستد ولیکن هرگاه کانال از قبل در اختیار دیگری باشد، ایستگاه بطور دائم به شنود کانال **نخواهد پرداخت**. در عوض [هرگاه ایستگاه کانال را مشغول تشخیص بدهد] به اندازه یک زمان تصادفی صبر کرده و پس از آن الگوریتم فوق را تکرار می‌کند. [تنها تفاوت این روش با روش قبلی در آنست که هرگاه کانال مشغول باشد ایستگاه مترصد آزاد شدن آن باقی نمی‌ماند و به اندازه یک زمان تصادفی کانال را به حال خود رها می‌کند و بدان گوش نمی‌دهد. -م] این الگوریتم طبعاً بهرهٔ کانال بهتری را ارائه می‌دهد [چرا که تصادمها در لحظهٔ آزاد شدن کانال کاهش می‌یابد] ولیکن تاخیر بیشتری نسبت به روش 1-Persistent CSMA دارد. [تاخیر ارسال]

آخرین پروتکل، p-Persistent CSMA نام دارد. این روش فقط برروی کانالهای زمان‌بندی‌شده (Slotted Time Channels) قابل اعمال است و بدین ترتیب عمل می‌کند: هرگاه ایستگاهی آمادهٔ ارسال شود ابتدا کانال را شنود می‌نماید؛ اگر کانال آزاد باشد فریم خود را با احتمال p ارسال می‌کند و یا به احتمال $q=1-p$ ارسال خود را تا فرا رسیدن برش بعدی زمان [اسلات بعدی] به تعویق می‌اندازد. یعنی حتی اگر یک اسلات خالی باشد ممکن است با احتمال p فریم خود را بفرستد یا با احتمال q به تعویق بیندازد. این فرآیند آنقدر تکرار می‌شود تا آنکه یا فریم ارسال شود یا آنکه ایستگاهی دیگر ارسال خود را آغاز نماید. در حالت دوّم [یعنی ایستگاهی دیگر موفق به ارسال شود] ایستگاه ناموفق، همانند وقتی که تصادم رخ داده عمل می‌کند یعنی به اندازهٔ یک زمان تصادفی صبر کرده و از نو شروع می‌نماید. اگر ایستگاه در همان ابتدا کانال را مشغول تشخیص بدهد تا اسلات بعدی صبر می‌کند و الگوریتم فوق را به اجرا می‌گذارد.

شکل ۴-۴ منحنی ظرفیت مفید (Throughput) کانال را بر مبنای حجم ترافیک تولید شده، برای سه پروتکل فوق و پروتکلهای Pure ALOHA و Slotted ALOHA نشان می‌دهد.

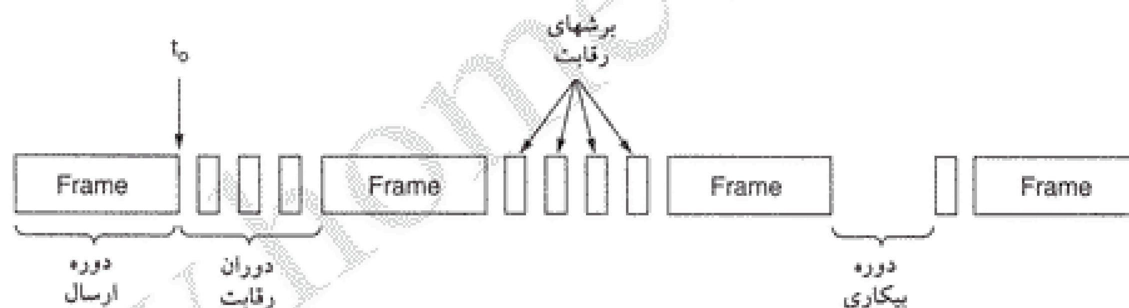


شکل ۴ـ۴. مقایسهٔ بهره‌وری کانال (ظرفیت مفید) بر حسب بار برای پروتکلهای گوناگون دسترسی تصادفی به کانال.

پروتکل CSMA با تشخیص تصادم

پروتکلهای Persistent & Nonpersistent CSMA به روشنی بهینه‌تر از ALOHA هستند زیرا مطمئناً اگر ایستگاهی کانال را مشغول تشخیص بدهد ارسال خود را آغاز نخواهد کرد. بهبود دیگر این روشها آنست که ایستگاهها به محض آنکه از وقوع تصادم آگاه شدند ارسال خود را نیمه‌کاره رها کنند. به عبارت دیگر هرگاه دو ایستگاه، کانال را آزاد احساس کرده و همزمان شروع به ارسال نمایند، تقریباً هر دوی آنها بلافاصله از وقوع تصادم مطلع خواهند شد. در چنین حالتی به محض کشف پدیده تصادم، ایستگاهها بجای ارسال کامل فریمهای آسیب دیده، به ارسال خود خاتمه می‌دهند. قطع سریع ارسال فریمهای آسیب‌دیده، در زمان و پهنای باند صرفه‌جویی خواهد کرد. چنین پروتکلی که اصطلاحاً CSMA/CD[1] نام دارد بطور گسترده در شبکه‌های محلی به کار گرفته شده است. به ویژه، این پروتکل مبنای شبکهٔ محلی و شناخته شده **اترنت** است، لذا ارزش آنرا دارد که اندک زمان بیشتری برای بررسی جزئیات آن صرف کنیم.

CSMA/CD نظیر بسیاری از پروتکلهای دیگر LAN از مدل مفهومی نشان داده شده در شکل ۴-۵ تبعیت می‌کند. در لحظه‌ای که با نماد t_0 مشخص شده، ایستگاهی ارسال فریم خود را به پایان رسانده است. در این لحظه ایستگاههایی که فریمی برای ارسال دارند ممکن است برای ارسال آن تلاش کنند. اگر دو یا چند ایستگاه بطور همزمان تصمیم به ارسال بگیرند تصادم رخ خواهد داد. وقوع تصادم را می‌توان با بررسی توان مصرفی یا اندازه‌گیری و مقایسهٔ پهنای پالس سیگنال دریافتی از کانال و مقایسهٔ آن با سیگنال ارسالی تشخیص داد.



شکل ۴ـ۵. CSMA/CD می‌تواند در یکی از سه وضعیت: «رقابت»، «ارسال» یا «بیکار» قرار داشته باشد.

پس از آنکه یک ایستگاه متوجه وقوع تصادم شد، ارسال خود را ناتمام رها کرده و از نو شروع می‌کند؛ (البته با این فرض که ایستگاه دیگری در این بینابین ارسال خود را آغاز ننماید.) بدین ترتیب مدل ارائه شده برای CSMA/CD ، شامل: (۱) چندین مرحلهٔ متناوب **«رقابت»** (Contention) (۲) بازه‌های ارسال و (۳) بازه‌های بیکاری خواهد بود. (بازه‌های بیکاری بدین معناست که تمام ایستگاهها بدلیل عدم نیاز به ارسال ساکت بوده‌اند.)

حال جزئیات الگوریتم رقابت را دقیقتر بررسی می‌نمائیم: فرض کنید دو ایستگاه بطور همزمان ارسال فریم خود را در زمان t_0 آغاز نمایند. چقدر طول می‌کشد تا متوجه شوند تصادم اتفاق افتاده است؟ پاسخ این سوال برای تعیین طول زمان رقابت و همچنین محاسبهٔ تاخیر و ظرفیت مفید کانال حیاتی است.

حداقل زمان لازم برای تشخیص وقوع تصادم، معادل زمانی است که طول می‌کشد تا سیگنال ارسالی از یک ایستگاه به ایستگاه دیگر منتشر شود. براساس این استدلال شاید شما تصور کنید که یک ایستگاه تا پس از زمانی معادل با زمان انتشار سیگنال برروی کلّ کابل از وقوع تصادم مطلع نشده و از تصرف کانال، مطمئن نخواهد بود. (منظورمان از **«تصرّف»** کانال آنست که ایستگاه‌های دیگر متوجه شوند که او در حال ارسال است و تداخلی پیش نیاید.) این نتیجه‌گیری صحیح نیست. به سناریوی زیر که در بدترین شرایط در نظر گرفته شده، دقت نمایید: زمان

1. Carrier Sense Multiple Access with Collision Detection

لازم برای انتشار سیگنال بین دورترین ایستگاه‌ها از یکدیگر را τ فرض کرده‌ایم؛ در لحظهٔ t_0 یک ایستگاه ارسال خود را آغاز می‌کند. در لحظهٔ $\tau - \varepsilon$ یعنی دقیقاً قبل از لحظه‌ای که سیگنال به دورترین ایستگاه می‌رسد، آن ایستگاه شروع به ارسال می‌کند. مشخصاً این ایستگاه وقوع تصادم را کشف می‌کند ولیکن سیگنال نویزی که در اثر تصادم تولید می‌شود تا زمان $2\tau-\varepsilon$ به ایستگاه مبداء باز نخواهد گشت. به عبارت دیگر در بدترین حالت، تا انقضای زمان 2τ پس از شروع ارسال ، ایستگاه مبداء نمی‌تواند از تصرف کانال مطمئن باشد. بهمین دلیل ما بازهٔ رقابت را همانند روش Slotted ALOHA مدل کرده‌ایم که در آن پهنای هر برش زمانی (اسلات) معادل 2τ می‌باشد. بر روی یک کابل کوآکسیال به طول ۱ کیلومتر، 2τ حدوداً معادل ۵ میکروثانیه است. برای سادگی فرض را بـر آن خـواهـیم گذاشت که هر برش زمان در برگیرنده تنها یک بیت [به طول 2τ] است و به محض آنکه کانال تصرف شد، ایستگاه می‌تواند با هر سرعت دلخواه ارسال خود را انجام بدهد ولیکن بدیهی است که با ارسال با نرخ $1bit/2\tau$ نخواهد بود!

درک این موضوع که فرآیند کشف تصادم به صورت آنالوگ انجام مـی‌شود اهـمیت دارد. سـخت‌افـزار هـر ایستگاه باید در حین ارسال به کابل گوش بدهد. اگر آنچه را که از کابل بازخوانی می‌کند با آنچه برروی کابل قرار داده، متفاوت باشد متوجه وقوع تصادم می‌شود. کشف تصادم مستلزم آنست که روش کدینگ سیگنال این امکان را فراهم بیاورد (زیرا مثلاً تصادم دو سیگنال صفر ولت قابل کشف نیست). به همین دلیل معمولاً از روشهای کدینگ خاص [مثل منچستر] استفاده می‌شود.

بدیهی است که ایستگاه فرستنده باید بطور مدام بر کانال نظارت کند و منتظر شنیدن سیگنال نـویزی کـه مشخص کنندهٔ وقوع تـصادم است بـاقی بـماند. بـهمین دلیـل CSMA/CD بـا یک کـانال مـنفرد ذاتاً سیستمی Half Duplex (دو طرفهٔ غیرهمزمان) است. در چنین سیستمی برای ایستگاه این امکان وجود ندارد که ارسال و دریافت فریمها را بطور همزمان انجام بدهد زیرا بخش گیرندهٔ آن حتی در خلال ارسال مشغول و در حال پیگیری بروز تصادم برروی کانال است. [البته در حین ارسال، گیرنده فقط آمادهٔ کشف تصادم به صورت آنالوگ است و داده‌های روی کانال، دریافت یا ذخیره نمی‌شوند.]

لازم است بدین نکتهٔ بدیهی اشاره کنیم که **پروتکل زیرلایهٔ MAC دریافت مطمئن فریمها را تضمین نمی‌کند** چراکه اگر حتی تصادمی رخ ندهد گیرندهٔ فریم ممکن است بدلائل گوناگون نتواند فریم را به درستی دریافت کند. (دلائلی مثل فقدان فضای بافر یا وقفه‌های گمشده)

۳-۲-٤ پروتکلهای بدون تصادم

اگرچه در روش CSMA/CD پس از آنکه یک ایستگاه بطور مطمئن کانال را تصرف کرد دیگر وقوع تصادم منتفی است ولیکن وقوع تصادمهای مکرّر در دورهٔ رقابت[1] کاملاً طبیعی است. این تـصادمها تـاثیر زیـانباری بـرروی کارآئی سیستم خواهد داشت؛ بالاخص زمانی که کابل، طولانی (و در نتیجه τ یا همان تاخیر انتشار بالاست) و یا فریمها کوتاه هستند، این تاثیر بسیار مخرّب است. در چنین حالاتی روش CSMA/CD قابل استفاده نخواهد بود. در این بخش چند پروتکل را بررسی می‌کنیم که در آنها مسئله رقابت حَل شده و بروز تصادم حتی در زمان رقابت منتفی است. اگرچه بسیاری از این روشها در حال حاضر برروی سیستمهای شناخته شده به کارگرفته نشده است ولیکن در اختیار داشتن پروتکلهایی با ویژگیهای عالی، برای به کارگیری در سیستمهای پیشرفتهٔ آینده یک حُسن محسوب می‌شود!

در پروتکلهایی که بررسی می‌شوند فرض را بر آن گذاشته‌ایم که دقیقاً N ایستگاه به کانال مشترک متصلند و

۱. Contention Period

هر ایستگاه آدرسی بین 0 تا N−1 دارد که درون سخت‌افزار ایستگاه حکّ شده است. این موضوع که ممکن است برخی از ایستگاه‌ها در برخی از مواقع غیر فعال و خاموش باشند چندان اهمیت ندارد. همچنین فرض کرده‌ایم که تاخیر انتشار قابل صرفنظر باشد. مسئلهٔ بنیادی هنوز باقی است: «پس از آنکه ارسال فعلی یک ایستگاه به پایان رسید کدامین ایستگاه حقّ دارد کانال را در اختیار بگیرد و فریم خود را ارسال نماید؟»

در این بخش نیز از مدل شکل ۴-۵ با برشهای گسستهٔ رقابت[1] استفاده می‌کنیم.

یک پروتکل مبتنی بر نشانه‌های بیتی (Bit Map)

در اولین پروتکل بدون تصادم که Basic Bit-Map Method نام دارد بازهٔ رقابت متشکل از دقیقاً N برش زمانی است. هرگاه ایستگاه شمارهٔ صفر، فریمی آمادهٔ ارسال داشته باشد در برش (اسلات) شماره صفر، بیت ۱ را برروی کانال می‌گذارد؛ هیچ ایستگاه دیگری حق ندارد در این برش چیزی برروی کانال بگذارد. فارغ از آنکه ایستگاه صفر چه کاری می‌کند ایستگاه شماره ۱ نیز این فرصت را دارد که در برش شماره ۱ با قرار دادن بیت ۱ برروی کانال تقاضای ارسال فریم بدهد. (البته اگر فریمی برای ارسال آماده داشته باشد.) بطور کلّی ایستگاه شماره j حق دارد در صورتی که فریمی جهت ارسال داشته باشد در برش شمارهٔ j با گذاشتن بیت ۱ برروی کانال، تقاضای خود را اعلام نماید. پس از آنکه کلّ N برش رقابت سپری شد تمام ایستگاه‌ها فهرست کاملی از ایستگاه‌های متقاضی ارسال، در اختیار دارند. پس از این لحظه، ایستگاههای متقاضی ارسال، به ترتیب شماره، ارسال فریمهای خود را آغاز می‌کنند. (شکل ۴-۶ را ملاحظه کنید)



شکل ۴-۶. پروتکل پایه مبتنی بر نشانه‌های بیتی (Basic Bitmap Protocol).

از آنجایی که تمام ایستگاه‌ها در هر لحظه می‌دانند که چه ایستگاهی حقّ ارسال دارد به هیچ وجه تصادم رخ نخواهد داد. پس از آنکه آخرین ایستگاه، فریم خود را ارسال کرد، (رخدادی که همه ایستگاهها بسادگی می‌توانند متوجه آن شوند) مجدداً دورهٔ رقابت، شامل N برش (یا بعبارتی N بیت) شروع می‌شود. هرگاه ایستگاهی پس از گذشت برش (اسلات) متعلق به او ، آمادهٔ ارسال شود فرصت را از دست داده و بایستی خاموش بماند تا دیگر ایستگاه‌ها شانس خود را آزموده و کارشان را انجام بدهند تا دور بعدی رقابت فرا برسد. پروتکلهایی همانند این روش که در آنها هر ایستگاه تقاضای خود را قبل از ارسال و به صورت فراگیر به اطلاع همه می‌رساند اصطلاحاً پروتکل‌های رزروسازی (Reservation) نامیده می‌شوند.

اجازه بدهید مختصراً به کارآئی این پروتکل بپردازیم. برای سادگی ، واحد زمان را معادل طول بیت هر برش رقابت در نظر گرفته‌ایم و طول هر فریم داده را معادل d واحد زمان فرض کرده‌ایم. [یعنی هر یک از برشهای رقابت را یک واحد و بر این اساس، طول فریم را d واحد زمان، تلقی کرده‌ایم.] در شرایطی که بار ایستگاهها پائین است، برشهای رقابت بطور متوالی تکرار می‌شوند و بدلیل عدم تقاضا برای ارسال فریم مکرراً خالی می‌مانند. حال وضعیت را از دیدگاه یک ایستگاه با شماره پائینی مثل صفر یا یک بررسی می‌کنیم. [بدون آنکه به عمومیت استدلال لطمه‌ای بخورد] وقتی چنین ایستگاهی آماده ارسال می‌شود باید تا رسیدن برش متناظر با شمارهٔ خودش صبر کند

1. Discrete Contention slots

در حالیکه ممکن است شمارهٔ برش فعلی یکی از شماره‌های میانی باشد. بطور «میانگین» ایستگاه مجبور است به اندازهٔ N/2 صبر کند تا دور فعلی خاتمه یافته و در برش متعلق به خودش تقاضا بدهد؛ سپس باید اندازهٔ N برش صبر کند تا تمام برشها به ترتیب بگذرند و او بتواند ارسال خود را آغاز کند. [چون نوبت او گذشته بطور میانگین باید N/2 برش دیگر صبر کند تا زمان ارسال او فرا برسد. بنابراین بطور میانگین زمانی معادل 1.5N معطل می‌شود.]

چشم‌انداز شانس ایستگاه‌های با شمارهٔ بالا روشنتر است. عموماً چنین ایستگاه‌هایی بطور میانگین مجبورند قبل از شروع به ارسال، N/2 برش صبر کند تا نوبت به اعلام تقاضا و سپس ارسال آنها برسد. ایستگاه‌های با شمارهٔ بالا به ندرت مجبورند که یک دور کامل صبر کنند تا دور بعدی فرا برسد. از آنجایی که ایستگاه‌های با شمارهٔ پائین باید بطور میانگین 1.5N و ایستگاه‌های با شمارهٔ بالا باید 0.5 N منتظر بمانند لذا میانگین این دو زمان N خواهد شد. با این استدلال، محاسبهٔ کارآئی کانال در بار پائین ساده است: میزان سرباری که باید برای هر فریم d بیتی متحمل شد N بیت است بنابراین این کارآئی کانال در بار پایین عبارتست از: d/(N + d)

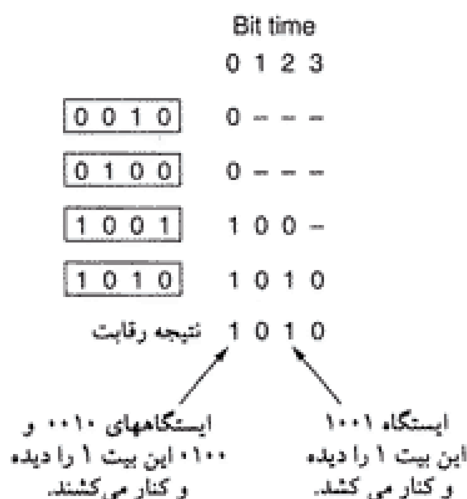
در بار بالا، یعنی وقتی تمام ایستگاه‌ها چیزی برای ارسال داشته باشند، N برش رقابت برای ارسال N تا فریم متوالی صرف می‌شود و این سربار به ازای هر فریم ۱ بیت خواهد بود [یعنی N بیت سربار برروی N فریم d بیتی سرشکن می‌شود لذا به ازای هر فریم d بیتی فقط یک بیت سربار تحمیل می‌شود.] بنابراین کارآئی کانال در بار بالا عبارتست از: d/(d+1) . میانگین تاخیر ارسال هر فریم نیز، معادل با زمان تاخیر در صفّ داخلی هر ایستگاه به اضافهٔ مقدار N.(d + 1)/2 (معادل تاخیر ارسال پس از آنکه فریم به سر صف داخلی ایستگاه می‌رسد) خواهد بود.

روش شمارش دودوئی معکوس (Binary Countdown)

مشکل پروتکل Bitmap آنست که به ازای هر ایستگاه یک بیت سربار تحمیل می‌شود فلذا شبکه قابل‌گسترش به مقیاس هزاران ایستگاه، نخواهد بود. می‌توان براساس آدرس دودوئی ایستگاه‌ها، فرآیند رزروسازی را به روش بهتری انجام داد: ایستگاهی که می‌خواهد کانال را به خدمت بگیرد شماره آدرس خود را به صورت دنباله‌ای از بیتها که از پرارزشترین آن شروع می‌شود برروی کانال منتشر می‌کند. فرض بر آنست که تمام آدرسها دارای طول یکسانی هستند. هر یک بیتهای آدرس پس از انتشار برروی کانال بطور منطقی با یکدیگر OR (Wired OR) می‌شوند. این روش که «**پروتکل شمارش دودوئی معکوس**» نام دارد برای اولین بار در سیستم Datakit (Fraser,1987) به کارگرفته شد. در این روش به صراحت فرض شده که تاخیر انتشار خط ناچیز است و تمام ایستگاه‌ها می‌توانند بطور همزمان بیتهای ارسالی را بر روی کانال را بشنوند.

برای پیشگیری از هرگونه برخورد، باید یک «قاعدهٔ داوری» (Arbitration Rule) اعمال شود: به محض آنکه ایستگاهی متوجه شود که بیت پرارزش صفر او که برروی کانال قرار گرفته با بیت ۱ بازنویسی شده از دور رقابت کنار می‌کشد. به عنوان مثال اگر ایستگاه‌های 0010 ، 0100 ، 1001 و 1010 بخواهند کانال را بدست بیاورند، همگی در اولین برش زمانی، بیت پرارزش خود را برروی کانال می‌گذارند، یعنی به ترتیب بیتهای 0 ، 0 ، 1 و 1 برروی کانال ظاهر می‌شود. این بیتها با یکدیگر Wired OR شده و نتیجهٔ آن ۱ خواهد بود. ایستگاه‌های 0010 و 0100 متوجه ظهور ۱ برروی خط شده و نتیجه می‌گیرند که ایستگاهی با شمارهٔ بالاتر برای تصرّف کانال رقابت می‌کند، لذا از دور فعلی خارج می‌شوند. در ادامه فقط ایستگاه‌های 1001 و 1010 به رقابت می‌پردازند.

بیت بعدی هر دو صفر است فلذا این دو ایستگاه بازهم ادامه می‌دهند. بیت بعدی روی کانال ۱ خواهد بود لذا ایستگاه 1001 کنار می‌کشد. در نهایت ایستگاه 1010 برندهٔ این رقابت خواهد بود چراکه دارای بالاترین شمارهٔ آدرس است. پس از پیروزی در این رقابت (که به صورت مزایده برگزار شده)، ایستگاه برنده می‌تواند فریم خود را ارسال کند و پس از آن دور بعدی «مزایده» آغاز می‌شود. عملکرد این پروتکل در شکل ۴-۷ به تصویر کشیده شده



شکل ۴-۷. پروتکل شمارش دودویی معکوس. خط تیره علامت سکوت ایستگاه است.

است. پروتکل فوق دارای این ویژگی است که ایستگاه‌های با شماره بالاتر اولویت بیشتری نسبت به ایستگاه‌های با شمارهٔ پائینتر دارند؛ این ویژگی براساس زمینه و نوع کار می‌تواند خوب یا بد باشد.

کارآئی کانال در این روش معادل $d/(d + \log_2 N)$ است. با اینحال اگر قالب فریم بگونه‌ای زیرکانه انتخاب شود که آدرس فرستندهٔ فریم، همان اولین فیلد باشد مقدار سربار $\log_2 N$ بیت نیز تلف نخواند شد و کارآئی کانال صددرصد است!

دو پژوهشگر به نامهای «مارک» و «وارد» (۱۹۷۹)، گونه‌ای از روش «شمارش دودوئی معکوس» را با استفاده از واسطهای موازی (Parallel Interfaces) به جای واسطهای سریال معرفی و تشریح کردند. [یعنی ارسال اطلاعات به جای سریال به صورت موازی انجام می‌شود.] آنها پیشنهاد کردند که برای آدرس‌دهی ایستگاه‌ها، [به جای شماره‌های ثابت] از شماره‌های مجازی استفاده شود. پس از آنکه ایستگاهی موفق به ارسال شد، به شمارهٔ تمام ایستگاه‌های قبل از آن، تا شمارهٔ صفر، یک واحد اضافه می‌شود تا در مرحلهٔ بعد، آنهایی که موفق به ارسال نشده‌اند اولویت بالاتری داشته باشند. [بدین ترتیب مسئله قبضه شدن کانال توسط یک ایستگاه حلّ خواهد شد.-م]

بعنوان مثال فرض کنید ایستگاه‌های C، H، D، A، G، B، E و F به ترتیب دارای اولویتهای ۷، ۶، ۵، ۴، ۳، ۲، ۱ و ۰ هستند و [از بین ایستگاه‌های متقاضی مثل D، A، B و F]، D موفق به ارسال فریم خود شده است. پس از ارسال، D به انتهای این فهرست رفته و ترتیب اولویتها به صورت C، H، A، G، B، E، F و D تغییر می‌کند. بدین ترتیب C به صورت مجازی ایستگاه شماره ۷ باقی می‌ماند ولی شمارهٔ A از ۴ به ۵ صعود و شماره D، از ۵ به صفر سقوط می‌کند. حال ایستگاه D فقط در صورتی قادر به در اختیار گرفتن کانال خواهد شد که هیچ ایستگاه دیگری بدان احتیاج نداشته باشد.

روش شمارش دودوئی معکوس نمونه‌ای است از یک پروتکل ساده، جالب و کارآمد که در انتظار کشف مجدّد، روزگار می‌گذراند و در آینده، منزلتی را برای خود خواهد یافت!!

۴-۲-۴ پروتکلهای با رقابت محدود

تاکنون دو استراتژی بنیادی را برای دستیابی به کانال در شبکه‌های مبتنی بر کابل، بررسی کرده‌ایم: روشهای رقابت و تصادم مثل CSMA و روشهای بدون تصادم. هر کدام از این استراتژیها را می‌توان بر اساس دو معیار مهّم رده‌بندی کرد: (۱) میزان تاخیر در بار پائین (۲) کارآئی و بهرهٔ کانال در بار بالا.

در بار پایین روشهای مبتنی بر رقابت (مثل ALOHA یا CSMA) ارجحتر است چراکه تاخیر ناچیزی دارند؛ ولیکن وقتی بار افزایش می‌یابد روشهای مبتنی بر رقابت ارزش خود را از دست می‌دهند زیرا سربار تحمیل شده در اثر مبارزه بر سر بدست آوردن کانال، به شدت افزایش می‌یابد. دقیقاً عکس این موضوع در مورد پروتکلهای بدون تصادم صادق است: در بار پائین تاخیر بالائی دارند ولی وقتی بار شبکه افزایش می‌یابد (برعکس پروتکلهای مبتنی بر رقابت)، کارآئی کانال رو به افزایش می‌گذارد.

روشن است که اگر بتوانیم دو ویژگی ممتاز این روشها را با هم ترکیب کنیم به یک پروتکل مطلوب خواهیم رسید که در بار پائین بروش رقابت عمل می‌کند (تا تاخیر کمی داشته باشد) ولی در بار بالا از روش بدون تصادم بهره می‌گیرد (تا کارآئی کانال افزایش یابد). این پروتکلها به نام «**پروتکلهای با رقابت محدود**» مشهور هستند و بررسی آنها، مطالعات ما را در مورد شبکه‌های مبتنی بر شنود سیگنال، به نتیجه نهائی می‌رساند.

پروتکلهای مبتنی بر رقابت که تاکنون بررسی کرده‌ایم «متقارن» (Symmetric) هستند بدین معنا که هر کدام از ایستگاه‌ها، با احتمال p تلاش می‌کند کانال را در اختیار بگیرد و برای تمام ایستگاه‌ها، این احتمال یکسان و مساوی p فرض شده است. می‌توان کارآئی کلّ سیستم را با انتساب احتمالات مختلف به ایستگاه‌های متفاوت، بهبود داد.

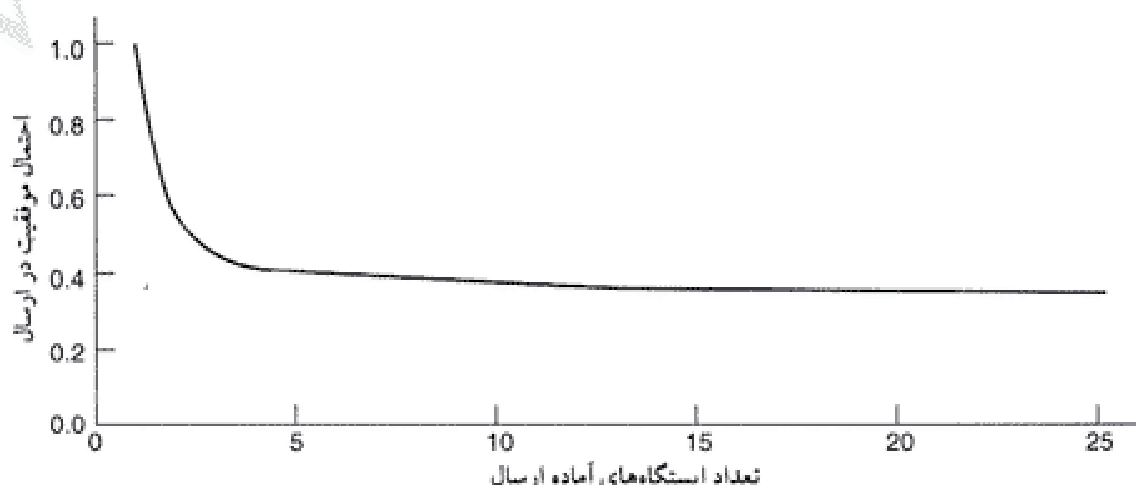
قبل از آنکه به پروتکلهای نامتقارن (Asymmtric) بپردازیم اجازه بدهید تا مروری اجمالی بر کارآئی شبکه در حالت متقارن داشته باشیم. فرض کنید تعداد k ایستگاه برای دسترسی به کانال رقابت می‌کنند و هر کدام در ابتدای یک برش زمان، با احتمال p اقدام به ارسال می‌نمایند. احتمال آنکه یکی از ایستگاه‌ها موفق به در اختیار گرفتن کانال در یکی از برشهای زمانی شود عبارتست از:

$$k.p.(1-p)^{k-1}$$

به منظور پیدا کردن مقداری بهینه برای p [به نحوی که احتمال فوق به حداکثر برسد] از رابطهٔ فوق بر حسب p مشتق گرفته و حاصل را مساوی صفر قرار می‌دهیم؛ سپس p را محاسبه می‌نمائیم. پس از محاسبه، بهترین مقدار p معادل 1/k بدست می‌آید. با قرار دادن 1/k به جای p در رابطهٔ فوق خواهیم داشت:

$$P_r[\text{موفقیت در ارسال با بالاترین احتمال}] = \left[\frac{k-1}{k}\right]^{k-1} \qquad \text{رابطه (۴-۴)}$$

منحنی این رابطه در شکل ۴-۸ ترسیم شده است. اگر تعداد ایستگاه‌ها کم باشد، احتمال موفقیت بالاست ولی به محض آنکه تعداد ایستگاه‌ها به ۵ یا بیشتر می‌رسد، احتمال موفقیت در ارسال، به مقدار تقریباً ثابت 1/e ، میل می‌کند.



شکل ۴-۸ منحنی احتمال موفقیت در تصرف کانال برای یک کانال متقارن.

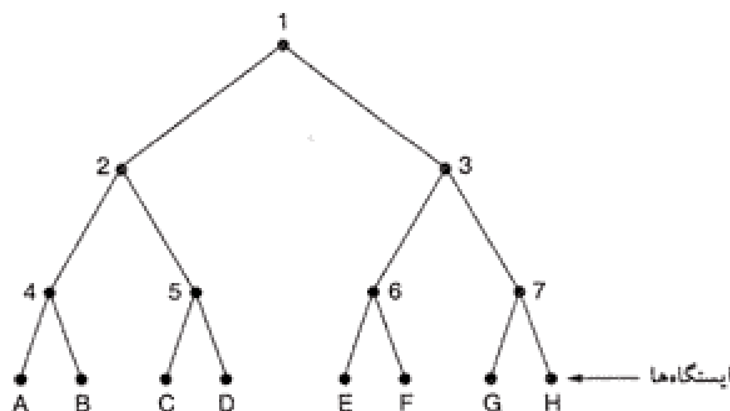
از شکل ۴-۸ بخوبی مشهود است که می‌توان احتمال موفقیت ایستگاه‌ها را با کاهش دادن حجم رقابت، افزایش داد؛ «پروتکلهای با رقابت محدود» سعی می‌کنند دقیقاً همین کار را انجام بدهند. این پروتکلها ابتدا ایستگاه‌ها را به چند گروه تقسیم می‌نمایند. (لزومی به از هم جدا بودن گروه‌ها نیست.) در برش زمانی شمارهٔ صفر، فقط اعضای گروه شمارهٔ صفر حق شرکت در رقابت را دارند. اگر یکی از آنها موفق شد، کانال را در اختیار می‌گیرد و فریم خود را ارسال می‌نماید. اگر برش زمانی شمارهٔ صفر خالی ماند یا تصادم رخ داد، اعضای گروه شمارهٔ ۱ در برش شمارهٔ ۱ رقابت می‌کنند و کار به همین ترتیب ادامه می‌یابد. با تقسیم‌بندی صحیح و مناسب ایستگاه‌ها به چند گروه، میزان رقابت در هر برش زمانی کاهش یافته و طبق منحنی شکل ۴-۸، شبکه در نزدیکی سمت چپ نمودار کار می‌کند.

نکته ظریف در اینجاست که ایستگاه‌ها را چگونه به برشهای زمانی مستقل منتسب نمائیم. [یعنی تعیین آنکه چه ایستگاه‌هایی در چه برش زمانی رقابت کنند. -م] یکی از حالات ویژه آنست که هر گروه صرفاً یک عضو بیشتر نداشته باشد. چنین انتسابی تضمین کنندهٔ آنست که هیچ تصادمی بوجود نخواهد آمد چرا که برای تصاحب هر برش زمانی فقط یک ایستگاه رقابت می‌کند. با چنین پروتکلهایی قبلاً برخورد کرده‌ایم (مثل روش شمارش دودوئی معکوس). حالت خاص دیگر آنست که در هر گروه دو ایستگاه قرار بگیرد. احتمال آنکه هر دوی آنها بطور همزمان و در یک برش زمانی بخواهند ارسال داشته باشند معادل p^2 است که برای مقادیر کوچک p، بسیار ناچیز خواهد بود. هر چه تعداد ایستگاه‌ها در هر گروه بیشتر باشد احتمال تصادم در برشهای زمانی متعلق به آن گروه افزایش خواهد یافت ولی در عوض تعداد بُرشهای زمانی لازم برای رقابت گروه‌ها کاهش می‌یابد. در محدودترین حالت تمام ایستگاه‌ها در یک گروه واحد قرار می‌گیرند؛ در این حالت عملکرد آن همانند روش Slotted ALOHA خواهد بود. بالطبع به روشی نیاز داریم که برشهای زمانی (Time Slots) را بطور پویا به ایستگاه‌ها اختصاص بدهد یعنی وقتی بار کم است تعداد زیادی ایستگاه در یک برش رقابت کنند ولیکن وقتی بار بالاست تعداد کمی ایستگاه (حتی یک ایستگاه) در هر گروه رقابت نمایند.

پروتکل پیمایش وفقی درخت (Adaptive Tree Walk)

یکی از روشهای بسیار ساده برای انجام عمل انتساب فوق‌الذکر، الگوریتمی است که توسط ارتش ایالات متحده در جنگ جهانی دوم برای آزمایش سربازان و تشخیص بیماری سیفلیس ابداع شد. (Dorfman,1943) ارتش، نمونهٔ خون N سرباز را می‌گرفت و بخشی از این نمونه‌های خون، درون یک لولهٔ آزمایش واحد ریخته و با هم مخلوط می‌شد. سپس این نمونهٔ مخلوط شده، مورد آزمایش قرار می‌گرفت و اگر هیچ آنتی‌بادی، در آن پیدا نمی‌شد تمام سربازان سالم تشخیص داده می‌شدند. در صورت تشخیص آلودگی در نمونهٔ خون، دو نمونهٔ دیگر از خونها تهیه می‌شد: نمونهٔ اوّل شامل مخلوطی از نمونهٔ خون سربازان 1 تا N/2 و نمونهٔ دومی از خون مابقی افراد. این فرآیند آنقدر تکرار می‌شد تا سربازان آلوده مشخص شوند.

برای نسخهٔ کامپیوتری این الگوریتم (Capetanakis,1979)، ساده‌تر آنست که ایستگاه‌ها را به عنوان برگهای یک درخت دودوئی فرض کنید (همانند آنچه که در شکل ۴-۹ می‌بینید). در اولین برش رقابت، (یعنی پس از خاتمهٔ ارسال یک ایستگاه و آزاد شدن کانال) تمام ایستگاه‌ها مجازند برای در اختیار گرفتن کانال رقابت کنند. اگر تصادمی به وقوع پیوست، در برش شمارهٔ ۱، فقط ایستگاه‌هایی که در پوشش گره ۲ از درخت واقعند به رقابت می‌پردازند. اگر یکی از آنها کانال را تصاحب و فریم خود را ارسال کرد، برش زمان بعدی، برای رقابت ایستگاه‌های تحت پوشش گره ۳ در نظر گرفته می‌شود ولیکن اگر بیش از یکی از ایستگاههای تحت پوشش گره ۲، بخواهند ارسال داشته باشند در همان برش شمارهٔ ۱ تصادم رخ می‌دهد و در برش شمارهٔ ۲ نوبت به رقابت ایستگاه‌های گره ۴ خواهد بود.



شکل ۴ـ۹. یک درخت برای هشت ایستگاه.

هرگاه در برش شمارهٔ ۰ تصادمی رخ بدهد، کلّ درخت به صورت «عمقی»[1] (Depth First) پیمایش می‌شود تا ایستگاهها به ترتیب شناسایی و تعیین موقعیت شده و ارسال خود را انجام بدهند. هر برش رقابت، به یک گره خاص در درخت تعلق دارد. اگر تصادمها تکرار شود، پیمایش و جستجو از چپ به راست و به صورت بازگشتی (Recursive) ادامه می‌یابد. اگر یکی از برشها خالی بماند یا فقط یک ایستگاه، بدون تصادم کانال را صاحب شود جستجو در آن گره خاتمه می‌یابد، چراکه تمام ایستگاه‌های آمادهٔ ارسال در آن گره، مشخص شده‌اند. (اگر بیش از یک ایستگاه در آن گره تمایل به ارسال می‌داشتند تصادم رخ می‌داد.)

وقتی بار شبکه بالا است به ندرت اتفاق می‌افتد که مسئله تخصیص کانال در همان برش رقابت شمارهٔ صفر که متعلق به گره ۱ است حلّ شود زیرا به احتمال زیاد بیش از یک ایستگاه آمادهٔ ارسال هستند. بدلیل مشابه این مسئله در برش ۲ و ۳ نیز حل نخواهد شد و رقابت به مراحل بعدی خواهد کشید. سوال آنست که بطور کلّی، جستجو از چه سطحی آغاز شود؟ روشن است که هر چه بار سنگینتر باشد، جستجو باید از سطوح پائین‌تر شروع شود. فرض را بر آن خواهیم داشت که هر ایستگاه تخمین خوبی از تعداد کلّ ایستگاه‌های آمادهٔ ارسال (که آنرا q می‌نامیم) در اختیار دارد و این تخمین را به روشی مثل نظارت بر ترافیک جاری شبکه بدست آورده است.

در ادامه اجازه بدهید سطوح درخت را شماره‌گذاری کرده و رأس آن یعنی گره شمارهٔ ۱ را سطح صفر بنامیم. بدین ترتیب، گره ۲ و ۳ در سطح یک قرار می‌گیرند و شماره گذاری سطوح بهمین نحو ادامه می‌یابد. دقت کنید که هرگره در سطح i، کسر $\frac{1}{2^i}$ از کلّ ایستگاه‌های شبکه را در بر می‌گیرد. اگر q ایستگاه آمادهٔ ارسال، بطور یکنواخت در گره‌ها توزیع شده باشند، میانگین تعداد ایستگاه‌های آمادهٔ ارسال که‌تحت پوشش گرهی خاص در سطح i قرار دارند، معادل $2^{-i}.q$ خواهد بود. بطور حسّی می‌توان انتظار داشت که سطح بهینه‌ای که جستجو باید از آن سطح آغاز شود همان سطحی است، تعداد متوسط ایستگاه‌های آماده در زیر هر گره ۱ باشد، یعنی همان سطحی که در آن $2^{-i}.q = 1$ است. با حلّ این معادله بدست می‌آوریم: $i = \log_2 q$.

دو پژوهشگر بنامهای Bertsekas و Gallager (1992)، نسخه‌های متعدد و پیشرفته‌تری از الگوریتم فوق را ابداع و جزئیات آنها را تشریح کرده‌اند. به عنوان مثال، حالتی را در نظر بگیرید که در آن ایستگاه‌های G و H [تحت پوشش گره ۷] تنها ایستگاه‌های آمادهٔ ارسال هستند. در گره ۱ تصادم رخ خواهد داد، در حالیکه وقتی در برش شماره ۲ نوبت به رقابت اعضای گره ۲ می‌رسد، آن برش خالی خواهد ماند. مجدداً در حین آزمایش گره ۳ تصادم پیش می‌آید؛ (تا اینجا می‌دانیم که دو یا چند ایستگاه در گره ۱ تمایل به ارسال دارند ولی قطعاً در گره ۲ واقع

---

۱. در مبحث ساختمان داده و الگوریتمها، پیمایش عمقی روشی برای پیمایش درخت محسوب می‌شود. ـم

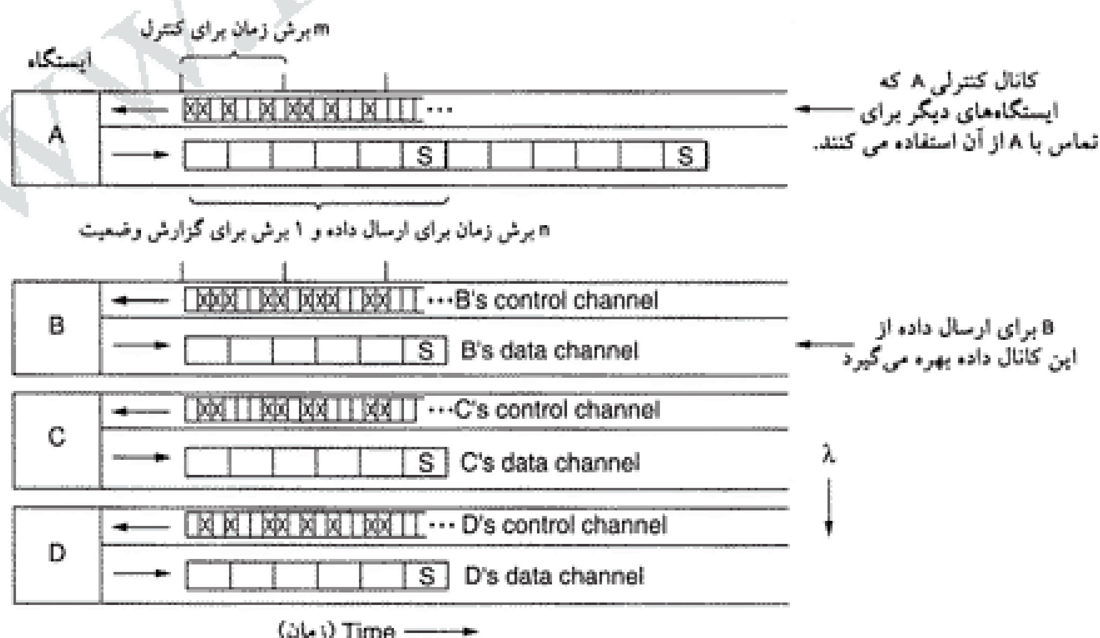
نشده‌اند). آزمایش گره ۳ ناکام مانده و به آزمون گره ۶ می‌انجامد. باز هم در آزمایش گره ۷ تصادم رخ می‌دهد و عاقبت در آخرین تلاش G موفق به ارسال خواهد شد. دو پژوهشگر فوق برای حلّ مشکلات این چنینی، راهکارهایی را معرفی کرده‌اند.

۴ـ۲ـ۵ پروتکلهای دسترسی چندگانه مبتنی بر تقسیم طول موج[۱]

یک راهکار متفاوت برای تخصیص کانال آنست که آنرا بروشی مثل FDM یا TDM (یا تلفیقی از هر دو) به چندین «زیرکانال» تقسیم کرده و آنها را در صورت نیاز به صورت پویا به ایستگاه‌ها تخصیص بدهیم. چنین الگویی، به طور رایج برروی شبکه‌های محلی با فیبرنوری کاربرد دارد تا با استفاده از طول موجهای مختلف (یعنی فرکانسهای متفاوت)، امکان محاوره و ارتباط همزمان ایستگاه‌ها فراهم آید. در این بخش یکی از این پروتکلها را بررسی خواهیم کرد.(Humblet et al., 1992)

ساده‌ترین راه برای ساختن یک شبکهٔ محلّی کاملاً نوری، آنست که شبیه به شکل ۲ـ۱۰، از ترویج کنندهٔ غیرفعّال با توپولوژی ستاره استفاده نمائیم. در حقیقت دو فیبرنوری منشعب شده از هر ایستگاه به یک استوانهٔ شیشه‌ای وارد شده‌اند. [در این استوانه تمام پرتوهای نوری با هم ممزوج شده، در هم می‌آمیزند.-م] یکی از تارهای فیبر نوری بعنوان خروجی و دیگری بعنوان ورودی ایستگاه از این استوانه منشعب می‌شوند. پرتوی نور خارج شده از یک ایستگاه، در درون استوانه منتشر شده و در ورودی تمام ایستگاهها قابل دریافت و تشخیص است. شبکه‌های نوری غیر فعال با توپولوژی ستاره می‌توانند صدها ایستگاه داشته باشند.

برای آنکه ارسال همزمان چندین ایستگاه میّسر باشد، طیف نوری به چندین کانال (باندهای مختلف با طول موج متفاوت) تقسیم می‌گردد. (شکل ۲ـ۳۱ را ببینید.) در این پروتکل که WDMA (دسترسی چندگانه مبتنی بر تقسیم طول موج) نامیده شده به هر ایستگاه دو کانال منتسب می‌شود: یک کانال با پهنای باند باریک که از آن به عنوان کانال کنترل، جهت هماهنگی (سیگنالینگ) با ایستگاه استفاده می‌شود و یک کانال با پهنای باند وسیع که برای ارسال فریم، در اختیار ایستگاه قرار می‌گیرد.



شکل ۴ـ۱۰. دسترسی چندگانه مبتنی بر تقسیم طول موج.

۱ – Wavelength Division Multiple Access Protocol

به نحوی که در شکل ۴-۱۰ دیده می‌شود هر کانال به چندین گروه برش زمانی مستقل (اسلات) تقسیم شده است. اجازه بدهید تعداد برشهای زمانی در کانال کنترل را m و تعداد برشهای زمانی کانال داده را n+1 فرض کنیم؛ n تا از برشهای کانال داده، برای ارسال فریم و آخرین آنها برای گزارش وضعیت خود ایستگاه، کاربرد دارد (در حقیقت، این برش برای گزارش برشهای آزاد در دو کانال به کار می‌آید.) در هر دو کانال [داده و کنترل] دنبالهٔ برشهای زمانی بطور دائم و متناوب تکرار می‌شوند و از برش شمارهٔ صفر آغاز می‌گردد. برش شمارهٔ صفر بگونه‌ای نشانه‌گذاری شده که دیگر ایستگاه‌ها بسادگی قادر به تشخیص آن هستند. تمام کانالها با استفاده از یک سیگنال ساعت سراسری سنکرون می‌شوند.

این پروتکل از سه ردهٔ متفاوت ترافیک حمایت می‌کند: (۱) ترافیک اتصال‌گرا با نرخ ثابت برای عملیاتی نظیر ارسال تصاویر ویدیوئی غیرفشرده (۲) ترافیک اتصال‌گرا با نرخ متغیر برای عملیاتی نظیر انتقال فایل (۳) ترافیک دیتاگرام مثل ارسال بسته‌های UDP.

در دو پروتکل اتصال‌گرا، نظریهٔ آن بوده که وقتی A بخواهد با B ارتباط برقرار کند ابتدا یک فریم کنترلی خاصّ به نام «فریم تقاضای اتصال» (CONNECTION REQUEST) در یک برش خالی از کانال کنترلی متعلق به B قرار می‌دهد. اگر B این تقاضا را پذیرفت مبادله، از طریق کانال داده انجام می‌شود.

هر ایستگاه دارای دو فرستنده و دو گیرنده به ترتیب زیر است:

۱. **یک گیرنده با طول موج ثابت برای گوش دادن به کانال کنترلی خودش**

۲. **یک فرستندهٔ قابل تنظیم برای ارسال بر روی کانالهای کنترلی دیگر ایستگاه‌ها**

۳. **یک فرستنده با طول موج ثابت برای انتقال فریمهای داده بر روی خروجی خودش**

۴. **یک گیرنده با طول موج قابل تنظیم برای انتخاب یکی از فرستنده‌های داده و گوش دادن به آن**

به عبارت دیگر هر ایستگاه مدام به کانال کنترلی خودش گوش فرا می‌دهد تا تقاضاهای ارتباط را دریافت کند ولیکن برای دریافت داده‌های دیگران، باید خود را با طول موج فرستنده تنظیم کرده و تطبیق بدهد. تنظیم طول موج به کمک ابزاری به نام «**اینترفیرومتر فابری-پرو یا ماخ-زندر**»[۱] انجام می‌شود که در حقیقت نوعی فیلتر نوری است که در آن به غیر از یک باند خاص از طول موجها، بقیه حذف می‌شوند.

حال ببینیم که ایستگاه A، چگونه یک کانال ردهٔ ۲ [یعنی ارسال اتصال‌گرا با نرخ متغیر] با B، مثلاً برای انتقال فایل ایجاد می‌کند. ابتدا A، گیرندهٔ دادهٔ خود را برروی طول موج ایستگاه B تنظیم می‌نماید و انتظار می‌کشد تا به یک «**برش گزارش وضعیت**»[۲] (Status Slot) برسد. این برش زمانی، مشخص می‌کند که کدامیک از برشها در کانال کنترلی، خالی و کدام تخصیص داده شده‌اند. بعنوان مثال، در شکل ۴-۱۰ می‌بینیم که از میان هشت برش کنترلی متعلق به ایستگاه B، شماره‌های ۰ و ۴ و ۵ آزاد و بقیه اشغال هستند. (علامت × بمعنای پربودن برش است.) ایستگاه A یکی از برشهای آزاد کانال کنترل (مثلاً ۴) را انتخاب کرده و پیام «**تقاضای اتصال**» (CONNECTION REQUEST) خود را در آن می‌گذارد. از آنجایی که B بطور مدام بر کانال کنترل خود نظارت دارد این تقاضا را می‌بیند و برش ۴ را به A انتساب می‌دهد. این انتساب در «برش گزارش وضعیت» از کانال دادهٔ متعلق به ایستگاه B، اعلام می‌شود. وقتی A این اعلام را دریافت می‌کند متوجه می‌شود که اتصالی یک طرفه برایش مهیا است. اگر A تقاضای اتصالی دوطرفه داشته باشد، ایستگاه B نیز باید همین روال را انجام بدهد.

ممکن است درست در زمانی که A سعی می‌کند برش شماره ۴ از کانال کنترل B را برای خود تصرف نماید، C

---

۱. Fabry-Perot or Mach-Zehnder Interferometer

۲. «برش گزارش وضعیت» (Status Slot)، آخرین برش در هر گروه از برشهای زمانی است.

نیز همین کار را بکند. در این حالت هیچکدام موفق نخواهند شد و با نظارت بر «برش گزارش وضعیت» متوجه این شکست می‌شوند. [چون پاسخی در «برش گزارش وضعیت» دریافت نمی‌کنند.] در این حالت هر یک به اندازهٔ یک مقدار زمان تصادفی صبر کرده و از نو شروع می‌نمایند.

در این لحظه، هر یک از طرفین روشی بدون اشکال برای ارسال پیامهای کنترلی کوتاه خود به طرف دیگر در اختیار دارند. برای انجام عملیات انتقال فایل، اکنون A می‌تواند پیغام کنترلی کوتاهی برای B مثلاً بدین مضمون بفرستد: «لطفاً داده‌های من را در برش داده شماره ۳ دریافت کنید؛ در این برش یک فریم داده برای شما ارسال شده است!» وقتی B این پیام کنترلی را دریافت می‌کند گیرنده خود را با طول موج کانال خروجی A تنظیم می‌کند. بسته به پروتکل لایه بالاتر، B نیز می‌تواند در صورت تمایل از چنین مکانیزمی برای برگرداندن پیام‌های تصدیق (ACK) استفاده نماید.

دقت کنید که مسئله دیگری نیز ممکن است بوجود بیاید و آن هم اینکه اگر A و C اتصالی با B داشته باشند و هر کدام بطور همزمان به B اعلام کند که به برش شماره ۳ مراجعه کند. B تقاضای یکی از آنها را بطور تصادفی انتخاب کرده و دیگری قادر به ارسال نخواهد شد.

برای ترافیک با نرخ ثابت، گونه دیگری از این پروتکل به کار گرفته می‌شود: وقتی A تقاضای ایجاد یک اتصال می‌کند، بطور لحظه‌ای فریمی را بدین مضمون برای B می‌فرستد: «آیا مجاز به ارسال دائم در برش شماره ۳ برای شما هستم؟» اگر B قادر به پذیرش چنین تقاضایی باشد (یعنی هیچ قرار قبلی برای این برش نگذاشته باشد)، یک اتصال با پهنای باند تضمین‌شده، ایجاد خواهد شد. در غیر این صورت A می تواند پیشنهاد دیگری را براساس خالی بودن برشهای دیگر ارائه بدهد. برای ترافیک رده سوّم (یعنی ترافیک دیتاگرام) نیز از گونهٔ متفاوتی استفاده می‌شود: به جای نوشتن پیغام «تقاضای اتصال» در یک برش کنترلی، پیام DATA FOR YOU IN SLOT 3 را بر روی کانال کنترل قرار می‌دهد (به مضمون آنکه در برش ۳ داده‌ای بر شما وجود دارد). اگر B در خلال برش شمارهٔ ۳ از کانال داده آزاد باشد، انتقال انجام می‌شود و در غیر این صورت فریم از دست خواهد رفت. بدین ترتیب نیاز به برقراری هیچ اتصالی نخواهد بود.

گونه‌های متفاوتی از این پروتکل قابل اجرا است. مثلاً به جای آنکه هر ایستگاه بطور مستقل برای خودش کانال کنترل داشته باشد، یک کانال کنترلی واحد بین همهٔ ایستگاه‌ها مشترک باشد. به هر ایستگاه نیز مجموعه‌ای از برشهای هر گروه منتسب می‌شود تا بتوان چندین کانال مجازی را برروی یک کانال فیزیکی واحد مالتی‌پلکس کرد. همچنین این امکان وجود دارد که فقط از یک فرستندهٔ قابل تنظیم و یک گیرندهٔ قابل تنظیم در هر ایستگاه استفاده شود و کانال واحد هر ایستگاه به m برش کنترلی و بدنبال آن n+1، برش داده تقسیم گردد. اشکال این روش آنست که ایستگاه‌های فرستنده مجبورند برای در اختیار گرفتن کانال، زمان بیشتری را منتظر بمانند و از طرفی دنبالهٔ فریمهای داده با فاصلهٔ بیشتری از هم ارسال می‌شوند چراکه فریم کنترلی مابین آنها قرار گرفته است.

تاکنون پروتکلهای WDMA بی‌شماری پیشنهاد و پیاده‌سازی شده که در بسیاری از جزئیات با هم متفاوت هستند. برخی از آنها دارای یک کانال کنترل واحد هستند در حالیکه برخی دیگر چندین کانال کنترلی دارند. بعضی از آنها تاخیر انتشار را به حساب آورده‌اند در حالیکه بعضی از آن چشمپوشی کرده‌اند. برخی از آنها زمان تنظیم [فیلتر گیرنده یا فرستنده] را به عنوان بخشی مشخص از مدل خود در نظر گرفته‌اند در حالیکه برخی از آن صرفنظر کرده‌اند. این پروتکلها از دیدگاه پیچیدگی پردازش، ظرفیت و بهرهٔ مفید و قابلیت گسترش نیز با یکدیگر متفاوتند. به سیستمی که از تعداد زیادی فرکانس استفاده می‌کند، اصطلاحاً DWDM (Dense Wavelength Division Multiplexing) اطلاق می‌شود. برای آگاهی بیشتر به مراجع ذیل مراجعه کنید:

(Bogineni et al., 1993; Chen, 1994; Goralski, 2001; Kartalopoulos, 1999;Levine & Akyidiz, 1995)

۴-۲-۶ پروتکلهای بی‌سیم برای شبکهٔ محلی

به موازات رشد تعداد دستگاه‌های محاسباتی و مخابراتی قابل حمل‌ونقل (همراه)، تقاضا برای اتصال آنها به دنیای خارج نیز افزایش یافته است. اگرچه حتی نخستین سری تلفنهای همراه امکان اتصال به تلفن دیگر را داشتند ولیکن کامپیوترهای کیفی، فاقد چنین قابلیتی بودند. چیزی نگذشت که مودمها به عنوان یک ابزار معمولی برروی کامپیوترهای همراه جا خوش کردند ولیکن برای برقراری ارتباط، می‌بایست سیم این کامپیوترها را به پریزهای دیواری تلفن وصل کرد. نیاز به سیم جهت اتصال به یک شبکهٔ ثابت ، بدین معناست که کامپیوترها قابل حمل‌ونقل هستند ولی «همراه» (Mobile) تلقی نمی‌شوند.

برای تحقق معنای واقعی «همراه بودن» ، کامپیوترهای کیفی نیاز به ارتباط از طریق سیگنالهای رادیوئی (یا مادون قرمز) داشتند. در چنین شرایطی، کاربران می‌توانند در حین گردش یا قایقرانی، نامه‌های خود را بخوانند یا نامه بفرستند. به گونه‌ای که در بخش ۱-۵-۴ اشاره شد، کامپیوترهای کیفی که از طریق سیگنالهای رادیوئی با یکدیگر مخابرهٔ اطلاعات می‌کنند ، شبکهٔ محلی بی‌سیم تلقی می‌شوند. اینگونه شبکه‌های محلی تفاوتهای عمده‌ای با شبکه‌های محلی رایج دارند و به پروتکلهای خاصی در زیرلایهٔ MAC نیازمندند. در این بخش برخی از این پروتکلها را بررسی خواهیم کرد. برای آگاهی بیشتر در مورد شبکه‌های محلی بی‌سیم مراجع (Geier, 2002; O'Hara and Petrick, 1999) مفیدند.

پیکربندی رایج برای شبکه‌های محلی بی‌سیم بدین نحو است که در یک ساختمان اداری تعدادی **ایستگاه ثابت** (که نقاط دسترسی -Access Point- نامیده می‌شوند) نصب می‌شود. تمام این ایستگاه‌های ثابت از طریق فیبرنوری یا سیم مسی به یکدیگر متصل هستند. اگر توان انتقال رادیویی ایستگاه‌های ثابت و کامپیوترهای کیفی بگونه‌ای تنظیم شود که محدوده‌ای حدود ۳ تا ۴ متر را پوشش بدهد هر اتاق در ساختمان نقش یک «سلول» را ایفاء خواهد کرد و کلّ ساختمان همانند سیستمهای معمولی تلفن سلولی (یعنی شبکهٔ تلفن همراه که در فصل ۲ بدان پرداختیم) عمل می‌کند ولی برخلاف سیستم تلفن سلولی ، هر سلول تنها دارای یک کانال واحد است که کل پهنای باند موجود را در برگرفته و تمام ایستگاه‌های درون آن سلول را پوشش خواهد داد. عموماً پهنای باند این شبکه بین ۱۱ تا ۴۵ مگابیت بر ثانیه است.

در توضیحات زیر برای ساده‌تر شدن بحث، فرض را بر آن گذاشته‌ایم که تمام فرستنده‌های رادیوئی، دارای برد ثابت و محدودی هستند. وقتی گیرنده‌ای در برد دو ایستگاه فعّال و در حال ارسال قرار بگیرد سیگنال دریافتی او ، مصدوم و بلااستفاده خواهد بود؛ درک این حقیقت که در اغلب شبکه‌های محلی بی‌سیم، تمام ایستگاه‌ها الزاماً در برد یکدیگر قرار ندارند، بسیار مهم است و منجر به پیچیدگی‌هائی خواهد شد. به علاوه هرگاه شبکهٔ محلی بی‌سیم در داخل ساختمان قرار گرفته باشد وجود دیوار بین ایستگاه‌ها می‌تواند تاثیر مخربی برروی برد هر ایستگاه داشته باشد.

یک روش ناشیانه برای تخصیص کانال در شبکهٔ محلی بی‌سیم، بکارگیری روش CSMA است: ایستگاه‌ها به کانال گوش بدهند و در صورتی که هیچ ایستگاه دیگری در حال ارسال نبود، انتقال فریم انجام شود. مشکلی که در این پروتکل وجود دارد [در حالیکه در شبکه‌های مبتنی بر کابل وجود ندارد] آنست که در محیط بی‌سیم ، تداخل امواج و تصادم فقط در گیرنده اهمیت دارد نه در فرستنده! برای آنکه به ماهیت این مشکل پی‌ببریم به شکل ۴-۱۱ که در آن چهار ایستگاه بی‌سیم ترسیم شده نگاهی بیندازید. اینکه کدام ایستگاه ثابت و کدامیک کامپیوتر کیفی [همراه] هستند، برای هدفی که در پیش داریم اهمیتی ندارد. برد رادیوئی ایستگاه‌ها به گونه‌ای است که A و B در



شکل ۴ـ۱۱. یک شبکهٔ محلی بی‌سیم (الف) A در حال ارسال (ب) B در حال ارسال.

برد یکدیگر هستند و سیگنال آنها می‌توانند با یکدیگر تداخل کرده، خراب شود. C نیز می‌تواند با B و D تداخل سیگنال داشته باشد ولی با A تداخل ندارد [چون در برد او نیست].

ابتدا فرض کنید که A در حال ارسال برای B، است: (شکل ۴-۱۱-الف) اگر C کانال را شنود کند هیچ چیزی از A نمی‌شنود چون در برد A نیست و بدین ترتیب به اشتباه نتیجه می‌گیرد که می‌تواند برای B ارسال داشته باشد. اگر C ارسال خود را آغاز نماید در ایستگاه B [با سیگنال ارسالی از A] تداخل کرده و فریم رسیده از A را نابود خواهد کرد. این مشکل که یک ایستگاه قادر نیست حضور یک رقیب را برروی کانال تشخیص بدهد، **«مشکل ایستگاه پنهان»** (Hidden Station Problem) نامیده می‌شود و از آنجا ناشی می‌شود که ایستگاه‌ها از هم دور بوده و سیگنال‌های یکدیگر را نمی‌شنوند.

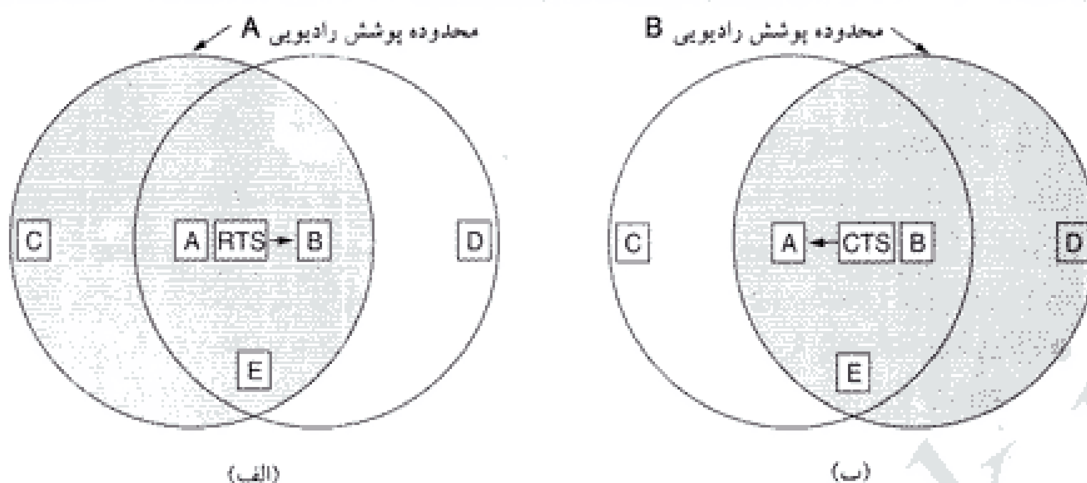
اکنون حالت برعکس شرایط فوق را بررسی می‌کنیم: به نحوی که در شکل ۴-۱۱-ب می‌بینید، B در حال ارسال برای A است. اگر C به شنود کانال بپردازد متوجه می‌شود که کانال اشغال است و به غلط نتیجه می‌گیرد که نباید برای D ارسال داشته باشد در حالیکه ارسال برای D هیچ اشکالی ندارد چرا که A و D در ناحیه‌ای دور از یکدیگر قرار گرفته‌اند و تصادمی پدید نخواهد آمد. این مشکل به نام **«مشکل ایستگاه آشکار»** (Exposed Station Problem) نامیده می‌شود.

مسئله اصلی آنست که یک دستگاه قبل از شروع به ارسال می‌خواهد بداند که آیا در پیرامون گیرنده، فعالیتی وجود دارد یا خیر؟ روش CSMA صرفاً می‌تواند در مورد فعالیت پیرامون فرستنده تشخیص خود را ارائه بدهد. به خاطر داشته باشید که بر روی سیم، سیگنالها به تمامی ایستگاه‌ها منتشر می‌شوند و طبعاً در آن واحد، فقط یک ایستگاه می‌تواند ارسال داشته باشد. در سیستمی که مبتنی بر امواج رادیوئی با برد کوتاه است چندین ایستگاه می‌توانند بطور همزمان ارسال داشته باشند البته مشروط به آنکه ایستگاه‌های مقصد، متفاوت و در محدودهٔ برد یکدیگر نباشند.

روش دیگری جهت اندیشیدن به این مشکل، تجسم اداره‌ای است که کارمندان آن کامپیوترهای کیفی بهمراه کارت شبکهٔ بی‌سیم در اختیار دارند. فرض کنید کارمندی مثل لیندا می‌خواهد پیامی را برای میلتون بفرستد. کامپیوتر لیندا محیط پیرامون خود را شنود کرده و چون فعالیتی را تشخیص نمی‌دهد ارسال را شروع می‌کند. غافل از آنکه در محل دفتر کار میلتون، تصادم رخ خواهد داد زیرا شخص ثالثی هم‌اکنون در حال ارسال برای اوست و از آنجا که دفتر کار آن شخص از لیندا دور بوده، فعالیت او تشخیص داده نشده است.

MACAW, MACA

یکی از اولین پروتکلهای طراحی شده برای شبکه‌های محلی بی‌سیم، پروتکل MACA (پروتکل دسترسی چندگانه با اجتناب از تصادم) است. (Karn,1990) ایدهٔ اصلی در این روش آنست که فرستنده به نحوی گیرنده را تحریک به ارسال یک فریم کوتاه برای ایستگاه‌های پیرامون خود کند تا آنهایی که در برد او هستند و این فریم کوتاه را می‌شنوند، از ارسال اطلاعات در خلال زمان دریافت فریم، خودداری کنند. روش MACA در شکل ۴-۱۲ نشان داده شده است.



شکل ۴-۱۲. پروتکل MACA. (الف) A در حال ارسال یک فریم RTS به B.(الف) B در حال ارسال فریم پاسخ CTS به A.

حال بررسی کنیم که چگونه A فریمی را برای B می‌فرستد. ایستگاه A کارش را با ارسال یک فریم کوتاه به نام RTS[1] برای B شروع می‌کند. (شکل ۴-۱۲-الف) درون این فریم کوتاه ۳۰ بایتی، طول کلّ فریم داده‌ای که قرار است در آینده ارسال گردد مشخص شده است. B در پاسخ، فریمی به نام CTS[2] ارسال می‌نماید. فریم CTS نیز طول فریم داده‌ای را که قرار است ارسال شود، مشخص می‌کند؛ (شکل ۴-۱۲-ب). پس از دریافت فریم CTS، ایستگاه A می‌تواند ارسال خود را آغاز نماید.

حال ببینیم ایستگاه‌هائی که این فریمها را می‌شنوند چه عکس‌العملی نشان می‌دهند. هر ایستگاهی که RTS را می‌شنود به A نزدیک است و باید به اندازه‌ای صبر کند تا پیام CTS، بدون تداخل و تصادم باز گردد. هر ایستگاه که CTS را می‌شنود، نزدیک به B (گیرندهٔ فریم داده) است و باید در خلال ارسال فریم کامل توسط A، ساکت بماند. از آنجا که طول فریم داده‌ای که در آینده قرار است ارسال شود، در فریم CTS مشخص شده، ایستگاه‌های شنوندهٔ CTS نیز می‌توانند زمان سکوت و انتظار، را تخمین بزنند.

در شکل ۴-۱۲ ایستگاه C در برد A است ولی در برد B نیست لذا RTS منتشره از A را می‌شنود ولی CTS ارسالی از B را نمی‌شنود. این ایستگاه مجاز است پس از آنکه به اندازهٔ زمان ارسال فریم CTS منتظر ماند، همزمان با ارسال فریم دادهٔ A، او هم به ارسال خود مشغول باشد. در سمت مقابل، D در برد B هست ولی در برد A قرار ندارد، فلذا RTS را نمی‌شنود در حالیکه CTS را می‌شنود. شنیدن CTS بدین معناست که ایستگاه در نزدیکی ایستگاه گیرنده واقع شده و ایستگاه‌های شنوندهٔ CTS باید ارسال خود را آنقدر به تعویق بیندازند تا ارسال فریم مورد انتظار به پایان برسد. ایستگاه E هر دو فریم کنترلی RTS و CTS را می‌شنود و همانند D بایستی منتظر بماند تا ارسال فریم خاتمه یابد.

با تمام این تمهیدات، باز هم وقوع تصادم محتمل است. به عنوان مثال اگر B و C هر دو بطور همزمان فریم RTS برای A بفرستند، تصادم رخ می‌دهد و فریم از بین خواهد رفت. در صورت بروز تصادم، فرستندهٔ ناموفق (یعنی فرستنده‌ای که به مدّت زمان مشخصی پس از ارسال RTS، فریم CTS دریافت نکند) به اندازهٔ یک زمان تصادفی صبر کرده و از نو تلاش می‌کند. الگوریتم انتظار و تکرار مجدّد به نام «الگوریتم عقبگرد نمایی» شهرت دارد و در مطالعهٔ شبکهٔ اترنت آنرا تشریح خواهیم کرد.

به منظور افزایش کارآیی پروتکل MACA، پژوهشگری به نام Bharghavan و گروه همکار او (۱۹۹۴) با

---

۱. تقاضا برای ارسال یا Request to Send

۲. مجوز ارسال یا Clear To Send

بررسی نتایج مطالعات شبیه‌سازی، اصلاحاتی را پیشنهاد کرده و پروتکل جدید را MACAW[۱] نامیدند. در بدو کار آنها متوجه شدند که اگر در لایهٔ پیوند داده ، پس از دریافت یک فریم سالم، پیام اعلام وصول (ACK) آن ارسال نگردد، طبعاً فریمهای از بین‌رفته، ارسال مجدّد نخواهد شد تا آنکه مدتی بعد و در لایهٔ انتقال، عدم وجود آنها کشف شود؛ این زمان انتظار بسیار طولانی و وقتگیر است و کارآیی پروتکل را بشدت کاهش خواهد داد. آنها این مشکل را با معرفی یک فریم جدید ACK که پس از دریافت موفق یک فریم داده ارسال می‌شود، حلّ کردند. همچنین آنها متوجه شدند که برخی از ویژگیهای CSMA باز هم در این شبکه قابل استفاده است؛ مثلاً برای آنکه یک ایستگاه ، همزمان با ایستگاه دیگری برای یک مقصد مشابه ، فریم RTS نفرستد ، بهتر است قبل از ارسال RTS کانال شنود شود؛ بدین ترتیب قابلیت شنود سیگنال حامل نیز به پروتکل اضافه شد. به علاوه آنها تصمیم گرفتند که الگوریتم «عقبگرد نمایی» را به جای آنکه برای هر ایستگاه به کار ببرند بطور مستقل برای یک استریم داده (که بین یک زوج مبداء و مقصد جریان دارد) اعمال نمایند. در آخر آنها به این پروتکل مکانیزمهائی افزودند تا ایستگاه‌ها با یکدیگر اطلاعاتی در خصوص ازدحام (Congestion) ردّ و بدل کنند. همچنین الگوریتم «عقب‌گرد نمایی» را بگونه‌ای تغییر دادند تا در خصوص مشکلات موقت، عکس‌العمل شدید از خودش نشان ندهد و بدین ترتیب کارآئی سیستم بهبود یابد.

۳ـ٤ اترنت

تا اینجا پروتکلهای تخصیص کانال را بصورت کلّی و اجمالی مطالعه کردیم؛ اکنون زمان آن فرا رسیده تا این قواعد را برروی سیستمهای واقعی و خصوصاً شبکه‌های LAN ، اعمال نمائیم. همانگونه که در بخش ۳.۵.۱ اشاره شد، IEEE تعدادی از شبکه‌های محلی و بین‌شهری را با نام IEEE 802 استانداردسازی کرده است. در فهرست شکل ۳۸ـ۱ دیدیم که برخی از این شبکه‌ها هنوز وجود دارند و برخی دیگر به تاریخ پیوسته‌اند. شاید آنهایی که به «نظریهٔ تناسخ روح» معتقدند بتوانند اینگونه بیندیشند که آقای چارلز داروین برگشته و به عنوان یکی از اعضای کمیتهٔ IEEE در حال حذف، پالایش و تکمیل زنجیرهٔ استانداردهاست!!! از بین استانداردهای باقیمانده از گذشته، می‌توان به 802.3 (اترنت) و 802.11 (شبکهٔ محلی بی‌سیم) اشاره کرد. هنوز زود است که در خصوص 802.15 (Blue tooth) و 802.16 (شبکهٔ بین شهری بی‌سیم) قضاوت کنیم؛ در این مورد می‌توانید به ویرایش پنجم این کتاب مراجعه کنید! لایهٔ فیزیکی و زیرلایهٔ MAC در شبکه‌های 802.3 و 802.11 تفاوت بنیانی دارند ولی این تفاوتها در زیرلایهٔ منطقی (تعریف شده در 802.2 LLC) ، همگرا خواهد شد و بدین ترتیب هر دوی آنها، واسط (Interface) مشابهی با لایهٔ شبکه دارند. [LLC فوقانی‌ترین زیرلایه از لایهٔ پیوند داده‌هاست و وظیفه دارد تفاوتها و ناهمگونیهای اجتناب‌ناپذیر زیرلایه‌های پایین را از دید لایهٔ شبکه مخفی نگاه دارد تا سرویسهایی که به این لایه ارائه می‌شود استاندارد و یکسان باشد. -م]

در خصوص اترنت در بخش ۱.۵.۳ توضیحاتی داده شده است؛ لذا آن مفاد را در اینجا تکرار نخواهیم کرد. در عوض، بر روی جزئیات تخصصی اترنت، پروتکلهای مرتبط با آن و پیشرفتهای اخیر در زمینهٔ اترنت سرعت‌بالا (گیگابیت اترنت) متمرکز خواهیم شد. از آنجایی که «اترنت» و IEEE 802.3 به غیر از دو تفاوت جزئی که به آنها اشاره خواهیم کرد، از بقیهٔ جهات یکسان هستند، بسیاری از افراد این دو شبکه را معادل هم در نظر گرفته و ما نیز این دو را یکی فرض خواهیم کرد. برای اطلاعات بیشتر در خصوص اترنت از مراجع زیر استفاده کنید:

Breyer and Riley, 1999; Seifert, 1998; and Spurgeon 2000)

۱. Multiple Access with Collision Avoidance for Wireless networks

۴-۳-۱ کابل‌کشی اترنت

از آنجایی که نام اترنت در اصل برگرفته از واژهٔ «اتر» است که به کابل اشاره دارد ما نیز توضیحات خود را از کابل شروع خواهیم کرد. در این شبکه چهار نوع کابل که فهرست آنها در شکل ۴-۱۳ آمده، رایج هستند:

نام کابل	نوع کابل	حداکثر طول قطعه	تعداد گره در هر قطعه	مزایا
10Base5	Thick coax	500 m	100	کابل اصلی و اولیه (از رده خارج)
10Base2	Thin coax	185 m	30	به هاب نیازی نیست
10Base-T	Twisted pair	100 m	1024	ارزان‌ترین سیستم
10Base-F	Fiber optics	2000 m	1024	بهترین انتخاب برای مابین ساختمانها

شکل ۴-۱۳. رایجترین انواع کابل‌کشی اترنت.

از منظر تاریخ، نخستین نوع کابل در اترنت ، 10Base5 بود که عموماً به «اترنت ضخیم» شهرت داشت. این کابل به شیلنگهای زردرنگ باغبانی شبیه است و هر ۲/۵ متر برروی آن علامتی گذاشته شده تا محل انشعاب تزریقی مشخص باشد. (البته طبق استاندارد 802.3 ، زردبودن کابل الزامی نیست ولی پیشنهاد شده است.) اتصال به کابل عموماً از طریق «**انشعاب تزریقی**» (Vampire Tap) انجام می‌شود که در آن یک سوزن به دقت در مرکز کابل کوآکسیال فرو می‌رود. نماد 10Base5 بدین معنی است که شبکه با نرخ 10Mbps کار می‌کند، از سیگنالینگ باند پایه (Baseband) بهره گرفته و طول حداکثر یک قطعه کابل ۵۰۰ متر است. گونهٔ دیگری کابل با نام 10Broad36 ، در باند وسیع (Broadband) طراحی شد ولی هیچگاه به بازار نیامد و عملاً از صحنه محو گردید. در صورتی که کانال از نوع کابل کوآکس باشد، عددی که بعد از کلمهٔ Base ظاهر می‌شود، حداکثر طول کابل را بر مبنای ۱۰۰ متر مشخص می‌کند.

نوع دوم کابل‌کشی اترنت 10Base2 یا «اترنت نازک» نام داشت که برخلاف کابل قبلی (که شبیه به شیلنگ باغبانی و غیرقابل‌انعطاف بود) به راحتی خم می‌شد. برای اتصال به این نوع کابل، بـه جـای استفاده از انشعاب تزریقی، می‌توان از کانکتورهای BNC معمولی و ایجاد یک اتصال بشکل T ، بهره گرفت. کانکتورهای BNC بسیار قابل‌اعتماد و کاربرد آنها ساده‌تر است. کابلهای اترنت نازک نیز، بسیار ارزان و نصب آن راحت است ولیکن حداکثر طول کانال به ۱۸۵ متر کاهش یافته و به هر قطعه کابل حداکثر می‌توان ۳۰ ایستگاه متصل کرد.

تشخیص طول بیش از اندازهٔ کابل ، انشعابات بد ، شل‌شدن اتصالات یا هـرگونه پـارگی در جـائی از آن، از اساسی‌ترین مشکلات این دو نوع کابل محسوب می‌شوند [زیرا بروز یکی از این اشکالات کلّ شبکه از کار خواهد انداخت.] به همین دلیل تکنیکی برای تشخیص این معایب ابداع شده است: یک پالس الکتریکی با شکل معمولی به درون کابل تزریق می‌شود. اگر این پالس به یک مانع [پارگی] یا به انتهای کابل برخورد کند، یک سیگنال بازتاب (Echo) تولید شده و باز خواهد گشت. با اندازه‌گیری دقیق زمان بین ارسال پالس و زمان دریافت بازتاب آن، پیدا کردن محل تقریبی عامل بازتاب [محل خرابی] کشف خواهد شد. به این روش اصطلاحاً «**بازتاب‌سنجی در حوزهٔ زمان**» (Time Domain Reflectometry) گفته می‌شود.

مشکلات تشخیص محل پارگی در کابل ، باعث شد که الگوی متفاوتی در سیم‌کشی این نوع شبکه به کارگرفته شود؛ در روش جدید هر ایستگاه یک کابل اختصاصی دارد که آنرا به یک هاب مرکزی متصل می‌کند. این هاب اتصال الکتریکی تمام ایستگاهها را از درون ، برقرار می‌سازد. معمولاً این سیمها، از نوع زوج سیم‌های معمولی خطوط تلفن هستند زیرا در ساختمان‌های اداری این سیم‌کشی از قبل وجود دارد و تعداد زیادی از این زوج سیمها بلااستفاده رها شده‌اند. به این روش سیم‌کشی اصطلاحاً 10BaseT گفته می‌شود. هابها ترافیک داده‌های ورودی را

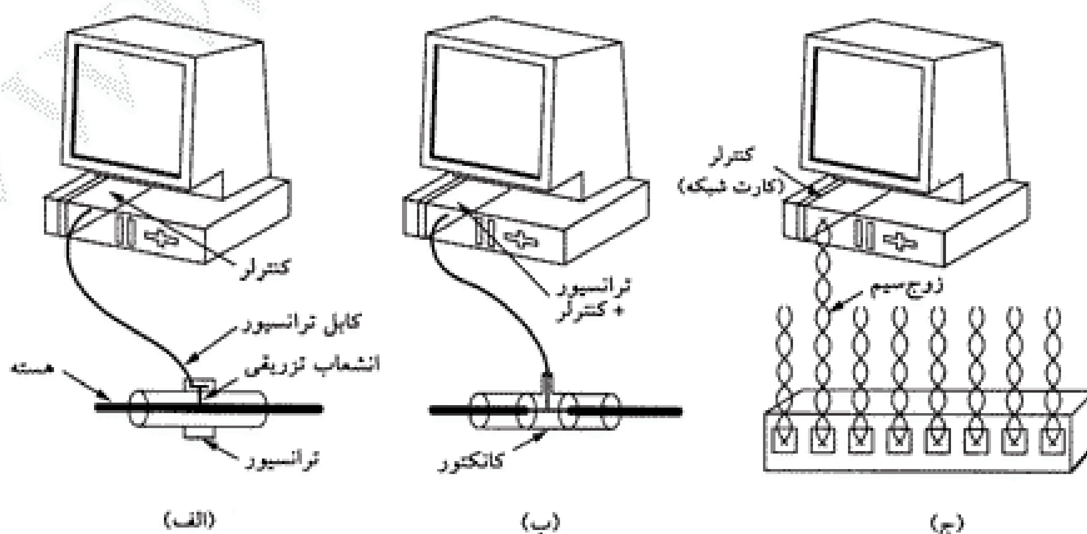
بافر نمی‌کنند [بلکه فقط ایستگاهها را به کانال مشترک متصل می‌نمایند]. در ادامهٔ همین فصل نسخهٔ پیشرفته‌تر این ایده (یعنی سوئیچها) را تشریح خواهیم کرد که قادرند ترافیک ورودی را بافر کنند.

این سه روش سیم‌کشی در شکل ۴-۱۴ به تصویر کشیده شده است. در سیم‌کشی 10Base5 ، مدار **ترانسیور**[۱] ، باید بدقت و با اطمینان به دور کابل اصلی، محکم شود تا اتصال آن با هستهٔ مرکزی کابل برقرار گردد. مدار ترانسیور دارای مدار الکترونیکی کوچکی است که حضور سیگنال حامل و بروز تصادم را تشخیص می‌دهد و در صورتی که متوجه بروز تصادم شود یک سیگنال خاص و غیرمعتبر بر روی کابل می‌گذارد تا مطمئن شود بقیه نیز از تصادم مطلع شده‌اند.

در 10Base5 ، یک کابل بنام ترانسیور (کابل اتصال) ارتباط کارت واسط ایستگاه و کابل اصلی را برقرار می‌کند. طول این قطعه کابل، می‌تواند تا ۵۰ متر باشد و درون آن ۵ جفت سیم زره‌دار (Shielded) وجود دارد. دو جفت از آنها برای داده‌های ورودی و خروجی از ایستگاه است. دو جفت نیز برای سیگنالهای کنترلی ورودی/خروجی به کارگرفته می‌شوند. زوج پنجم که ممکن است از آن استفاده نشود برای تغذیهٔ مدار الکترونیکی ترانسیور کاربرد دارد. همچنین، بعضی از ترانسیورها اجازه می‌دهند تا حداکثر هشت کامپیوتر نزدیک به هم، بدانها متصل شود تا تعداد ترانسیورهای مورد نیاز کاهش یابد.

کابل ترانسیور از یک طرف به کارت شبکه در درون کامپیوتر متصل است؛ کارت شبکه شامل یک تراشهٔ کنترلر است که ارسال یا دریافت فریم به ترانسیور را بر عهده دارد. این کنترلر وظیفه دارد داده‌ها را در قالب یک فریم مناسب سازماندهی کند. همچنین محاسبهٔ کدهای کشف خطا برای فریمهای خروجی و ارزیابی صحت فریمهای ورودی بر عهدهٔ همین کنترلر است. برخی از این تراشه‌ها دارای مقداری بافر هستند تا فریم‌های ورودی را جهت ارسال به صف نمایند؛ همچنین قادرند داده‌ها را بروش DMA به حافظه اصلی کامپیوتر منتقل کنند و در ضمن برخی از عملیات مدیریت شبکه را نیز انجام می‌دهند.

در سیم‌کشی 10Base2 ، اتصال کامپیوتر با کابل بکمک یک کانکتور BNC معمولی (از نوع T-شکل) انجام می‌شود. در این نوع سیم‌کشی، بخش الکترونیکی ترانسیور برروی بُرد کنترلر قرار گرفته است و بدین ترتیب هر ایستگاه ترانسیور خود را دارد.



شکل ۴-۱۴. سه روش کابل‌کشی اترنت (الف) 10Base5 (ب) 10Base2 (ج) 10Base-T.

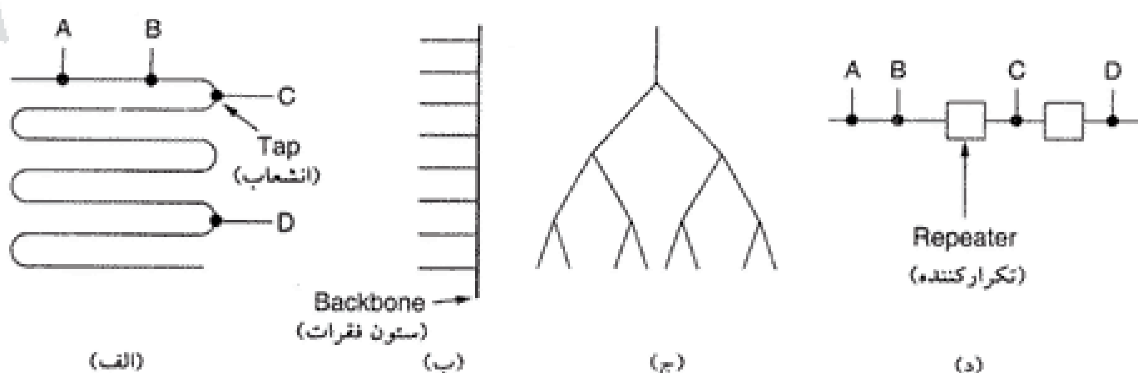
۱. ترانسیور (Tranciever) قطعه‌ای الکترونیکی است که اتصال کابَل نازک متصل به ایستگاه و کابل ضخیم اصلی را برقرار می‌کند. -م

در ساختار 10Base-T ، هیچ کابلی مشترکی وجود ندارد بلکه فقط یک **هاب** (جعبه‌ای پر از مدارات الکترونیک) وجود دارد که تمام ایستگاه‌ها با یک کابل اختصاصی (غیرمشترک) بدان متصل می‌شوند. حذف یا اضافهٔ یک ایستگاه بدین ساختار بسیار ساده و هرگونه پارگی در کانال به راحتی قابل کشف است. اشکال ساختار *10Base-T آنست که طول کابل متصل به هاب، حداکثر می‌تواند صد متر و در صورت استفاده از کابل باکیفیت و گرانی مثل Cat 5 ، حداکثر دویست متر باشد.* علیرغم این محدودیت، ساختار 10Base-T بدلیل سادگی نصب و پشتیبانی و همچنین استفاده از سیم‌کشی موجود ساختمانها، بسیار رایج شده است. نسخه سریعتر 10Base-T یعنی 100Base-T در ادامهٔ همین فصل تشریح خواهد شد.

روش چهارم کابل‌کشی، 10Base-F است که در آن از فیبرنوری بهره گرفته شده است. این گزینه بدلیل هزینهٔ بالای اتصالات و پایانه‌های[1] مورد نیاز، بسیار گران است ولی در عوض ایمنی بسیار بالائی در مقابل نویز دارد و انتخاب مناسبی برای کابل‌کشی بین ساختمانها یا هابهای دور از هم، به شمار می‌رود. در این روش، رشته‌هائی با طول کیلومتر نیز مجاز هستند. همچنین در این روش ضریب امنیت اطلاعات بسیار بالاست چراکه انشعات گرفتن از این کانال [و استراق سمع داده‌ها] بسیار دشوارتر سیم‌های مسی است.

شکل ۴-۱۵ روشهای مختلف کابل‌کشی ساختمان را نشان می‌دهد. در شکل ۴-۱۵-الف یک کابل واحد، اطاق به اطاق کشیده شده و هر ایستگاه در نزدیکترین نقطه بدان متصل شده است. در شکل ۴-۱۵-ب یک رشتهٔ عمودی در نقش ستون فقرات از طبقهٔ همکف تا بام کشیده شده و در هر طبقه کابلهای افقی از طریق تقویت کننده‌های خاص (بنام تکرارکننده) به این ستون فقرات متصل شده‌اند. در برخی از ساختمانها کابلهای افقی، نازک و کابل ستون‌فقرات، از نوع ضخیم انتخاب می‌شود. رایجترین ساختار، توپولوژی درختی است که در شکل ۴-۱۵-ج دیده می‌شود.

برای هر یک از روشهای کابل‌کشی اترنت، طول هر قطعه کابل نباید از یک مقدار حداکثر تجاوز کند. برای شبکه‌های وسیع می‌توان شبیه به شکل ۴-۱۵-د از چندین قطعه کابل که توسط «تکرارکننده» بهم متصل شده‌اند، استفاده کرد. تکرارکننده یک ابزار در لایهٔ فیزیکی است داده‌ها را دریافت، تقویت (بازتولید) و مجدداً ارسال می‌کند. از دیدگاه نرم‌افزار، مجموعه‌ای از قطعات کابل که از طریق تکرارکننده به هم متصل شده‌اند هیچ تفاوتی با یک شبکه با کابل یکپارچه ندارد (مگر اندکی تاخیر اضافی که توسط تکرارکننده‌ها تحمیل می‌شود.) یک سیستم ممکن است از چندین قطعه کابل و چند تکرارکننده تشکیل شده باشد ولیکن فاصله بین هر دو ترانسیور نبایستی از ۲/۵ کیلومتر تجاوز کند و در مسیر بین هر دو ترانسیور نباید بیش از ۴ تکرارکننده موجود داشته باشد.



شکل ۴-۱۵ توپولوژیهای مختلف کابل (الف) خطی (ب) ستون فقرات (ج) درختی (د) چندبخشی.

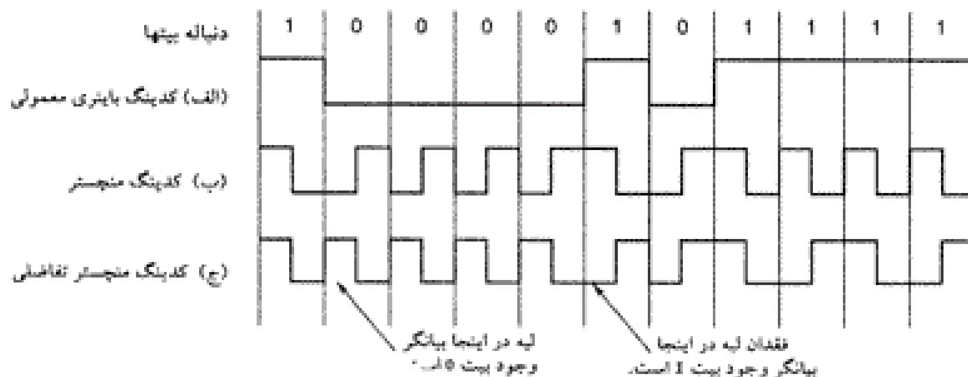
1. Terminators

۲ـ۳ـ٤ کدینگ منچستر

در هیچیک از نسخه‌های مختلف اترنت، از روش معمولی کدینگ (یعنی صفر ولت برای بیت 0 و ۵ ولت برای بیت 1) استفاده نشده است چراکه منجر به برخی اشکالات و ابهام [در دریافت بیت‌ها] خواهد شد: مثلاً اگر ایستگاهی رشته بیت 0001000 را ارسال کند، ایستگاه‌های دیگر، ممکن است به غلط آنرا 10000000 یا01000000 تشخیص بدهد زیرا ایستگاه نمی‌تواند تفاوت بین آزاد بودن خط (معادل صفر ولت) و بیت صفر (باز هم معادل صفر ولت) را تشخیص بدهد. این مسئله را می‌توان با در نظر گرفتن ۱+ ولت برای بیت یک و ۱- ولت برای بیت صفر، حل کرد ولیکن باز هم مشکل اختلاف فرکانس نمونه‌برداری از سیگنال (نسبت به فرکانس اصلی) باقی می‌ماند. تفاوت در سرعت سیگنال ساعت فرستنده و گیرنده باعث خواهد شد که این دو از حالت سنکرون خارج شوند و در اینصورت محدودهٔ هر بیت قابل تشخیص نخواهد بود؛ بالاخص وقتی یک دنباله طولانی پی‌درپی صفر یا دنبالهٔ یک پشت سرهم ارسال شود.روش معمولی کدینگ را در شکل ۴-۱۶-الف می‌بینید.

به روشی نیاز است تا گیرنده بتواند به درستی شروع، پایان یا وسط هر بیت را بدون نیاز به یک سیگنال ساعت خارجی تشخیص بدهد. دو روش به نامهای «کدینگ منچستر» و «کدینگ منچستر تفاضلی» دارای چنین قابلیتی هستند. در روش منچستر هر بیت از لحاظ زمانی به دو نیم بیت تقسیم می‌شود: برای ارسال بیت ۱، در نیم بیت اوّل، ولتاژ بالا (High) و در نیمهٔ دوم ولتاژ پائین (Low) قرار داده می‌شود. برای بیت صفر برعکس عمل می‌شود: در نیمهٔ اوّل ولتاژ پائین و در نیمهٔ دوّم ولتاژ بالا ارسال می‌گردد. این روش اطمینان می‌دهد که در وسط هر بیت یک لبه (Transition) وجود دارد. اشکال روش منچستر آنست که در مقایسه با روش معمولی، به پهنای باند دو برابر نیاز دارد چرا که طول هر پالس نصف طول یک بیت است. [در حقیقت برای هر بیت ۲ پالس ارسال می‌شود]. به عنوان مثال برای ارسال داده با سرعت ده‌مگابیت برثانیه، سیگنال تولیدی باید با نرخ ۲۰ میلیون بار در ثانیه، تغییر سطح ولتاژ داشته باشد. روش کدینگ منچستر در شکل ۴-۱۶-ب نشان داده شده است.

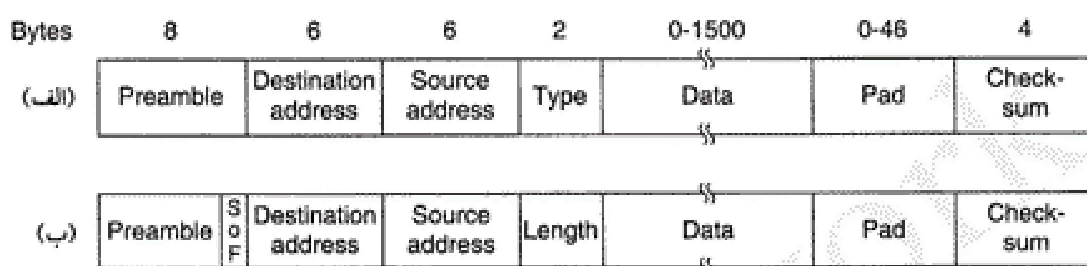
به گونه‌ای که در شکل ۴-۱۶-ج می‌بینید روش منچستر دیفرانسیلی نیز گونه‌ای از روش منچستر معمولی است. عدم وجود هرگونه لبه در ابتدای هر بیت(ابتدای Bit Time) نشان دهندهٔ بیت ۱ است و وجود لبه در ابتدای هر بیت، بیت صفر را مشخص می‌کند. در هر دو حالت، قطعاً یک لبه در میانهٔ هر بیت وجود دارد. روش منچستر تفاضلی به ابزارهای پیچیده‌تری نیازمند است ولیکن در عوض ایمنی بیشتری در مقابل نویز از خود نشان می‌دهد. تمام سیستمهای اترنت برای سادگی از روش کدینگ منچستر معمولی استفاده کرده‌اند و در آنها سطح بالای سیگنال (حالت High) مقدار ۰/۸۵+ ولت و سطح پایین سیگنال (حالت Low) مقدار ۰/۸۵- ولت و مقدار DC سیگنال نیز صفر است. در اترنت از روش منچستر تفاضلی استفاده نمی‌شود ولیکن در برخی از شبکه‌های محلی دیگر (مثل IEEE 802.5 Token Ring) از آن استفاده شده است.



شکل ۴ـ۱۶. (الف) کدینگ باینری معمولی (ب) کدینگ منچستر (ج) کدینگ منچستر تفاضلی.

۴-۳-۳ پروتکل زیرلایهٔ MAC در اترنت

قالب اصلی فریم DIX (پیشنهاد شرکت‌های Xerox ، Intel ، DEC) در شکل ۴-۱۷-الف نشان داده شده است. هر فریم با هشت بایت Preamble (دیباچه) شروع می‌شود که تمام بایتها دارای الگوی 10101010 هستند. کدینگ منچستر این الگوی هشت بایتی، بمدت ۶/۴ میکروثانیه یک سیگنال ساعت مربعی ده‌مگاهرتز تولید می‌کند تا بکمک آن گیرنده بتواند سیگنال ساعت خود را با سیگنال ساعت فرستنده سنکرون کند. گیرندگان موظفند با بهره‌گیری از ویژگی کدینگ منچستر تا انتهای فریم، سنکرون باقی بمانند و محدودهٔ بیتها را به درستی تشخیص بدهند.



شکل ۴-۱۷. قالب فریم (الف) اترنت DIX (ب) اترنت IEEE 802.3.

در این فریم دو آدرس تعریف شده است: یکی برای آدرس مقصد و یکی برای آدرس مبداء. طبق استاندارد، استفاده از آدرسهای ۲ یا ۶ بایتی مجاز است ولیکن در مشخصات معرفی شده برای استاندارد 10Mbps ، آدرسها صرفاً شش بایتی هستند.[۱] اگر مقصد فریم یک ایستگاه واحد باشد، پرارزشترین بیت آدرس مقصد، صفر و اگر مقصد فریم یک گروه از ایستگاهها باشد، ۱ است. آدرسهای گروهی این امکان را فراهم می‌آورند تا چندین ایستگاه بتوانند به یک آدرس واحد گوش بدهند. وقتی فریمی به یک آدرس گروهی ارسال می‌شود تمام ایستگاه‌های گروه قادرند آنرا دریافت نمایند. ارسال فریم برای یک گروه، اصطلاحاً چندپخشی (Multicast) نامیده می‌شود. هرگاه تمام بیتهای آدرس مقصد در یک فریم، ۱ باشند به این آدرس «پخش فراگیر» (Broadcast) گفته می‌شود. فریمی که تمام بیتهای فیلد آدرس مقصد آن ۱ است توسط همهٔ ایستگاه‌های شبکه دریافت خواهد شد. تفاوت بین ارسال «چندپخشی» و «فراگیر» آنقدر مهم است که ارزش تکرار مجدد را دارد: یک فریم چندپخشی برای یک گروه انتخابی از ایستگاه‌ها در شبکهٔ اترنت ارسال می‌شود در حالیکه فریم فراگیر، برای تمام ایستگاه‌های شبکه ارسال می‌گردد. ارسال چندپخشی، انتخابی و منعطف است ولیکن نیاز به مدیریت گروه‌ها دارد. در عوض ارسال فراگیر غیرمنعطف و غیرانتخابی است ولیکن به مدیریت گروه نیازی ندارد.

یکی دیگر از ویژگیهای جالب آدرس‌دهی در اترنت آنست که بیت چهل و ششم (مجاور بیت پرارزش)، سراسری یا محلّی بودن آدرسها را مشخص می‌کند. آدرسهای محلّی، آدرسهائی هستند که توسط مسئول شبکه تعیین شده و در خارج از شبکهٔ محلی هیچ معنا و ارزشی ندارند. در مقابل آدرسهای سراسری بطور خاص و مرکزی توسط IEEE اختصاص داده می‌شوند تا این اطمینان حاصل شود که هیچ دو ایستگاهی در کلّ دنیا دارای آدرس سراسری یکسان نیستند. ۴۶ بیت باقیمانده (۴۶ = ۲ - ۴۸) فضائی معادل 7×10^{13} آدرس سراسری ایجاد می‌کند. ایدهٔ اصلی آنست که هر ایستگاه به صورت یکتا و منحصربه فرد آدرس‌دهی شود. تعیین موقعیت و مشخص کردن ایستگاه مقصد، برعهدهٔ لایهٔ شبکه گذاشته شده است.

در ادامه، فیلد Type تعریف شده که به گیرندهٔ فریم تفهیم می‌کند که با این فریم چه کاری بکند. وقتی که

---

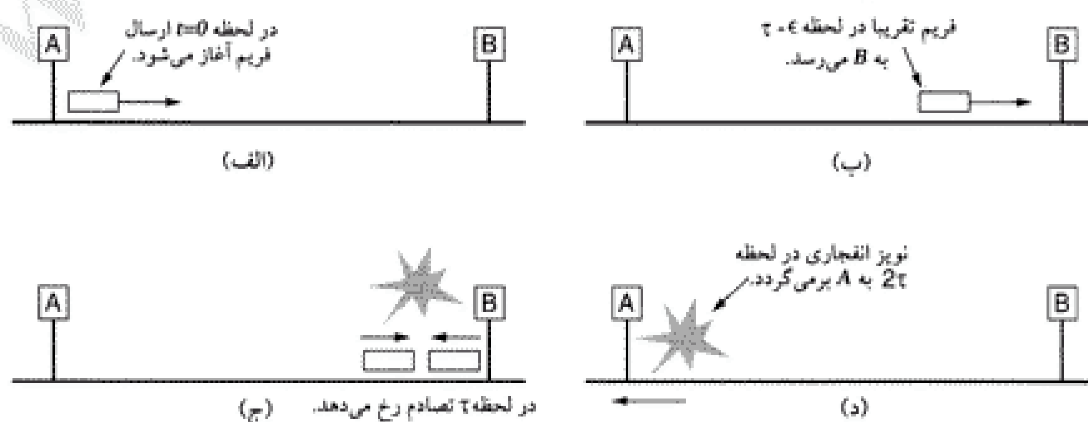
۱. امروزه در هیچیک از گونه‌های مختلف اترنت آدرسهای دوبایتی وجود خارجی ندارند. -م

چندین پروتکل لایهٔ شبکه، بطور همزمان بر روی یک ماشین واحد اجرا شده باشند، هستهٔ سیستم عامل پس از دریافت فریم، براساس این فیلد تصمیم می‌گیرد که آنرا به دست چه پروسه‌ای بسپارد. محتوای فیلد Type، پروسه‌ای که باید این فریم را تحویل بگیرد، مشخص می‌کند.

در ادامه، «فیلد داده» قرار می‌گیرد که گنجایش آن حداکثر ۱۵۰۰ بایت است. محدودیت ۱۵۰۰ بایتی، در زمان ارائهٔ استاندارد DIX ، بدلخواه انتخاب شد و انگیزهٔ اصلی طراحان از این انتخاب، صرفه‌جویی در میزان حافظهٔ لازم برای کارتهای شبکه بوده است زیرا در آن سالها (۱۹۷۸) حافظهٔ RAM گران تمام می‌شد.

گذشته از آنکه طول فریم دارای سقف حداکثر است، حداقل طول فریم نیز محدودیت دارد. گاهی ارسال فریمهای بدون داده، مفید خواهد بود ولی استفاده از چنین فریمهایی می‌تواند مشکل‌آفرین باشد. وقتی تصادم کشف می‌شود، فرستنده از ارسال باقیماندهٔ فریم جاری صرفنظر می‌کند و بدین نحو دنباله‌ای از بیتها (که در حقیقت قطعهٔ ابتدایی فریم است) روی کانال رها می‌شود. برای تشخیص فریمهای معتبر از فریمهای آشغال ، در اترنت نیاز است که طول فریم‌های معتبر حداقل ۶۴ بایت باشد که این ۶۴ بایت از ابتدای آدرس مقصد تا انتهای فیلد Checksum در نظر گرفته می‌شود. اگر بخش دادهٔ فریم از ۴۶ بایت (۱۸-۶۴) کمتر باشد در فیلد Pad آنقدر صفر اضافه می‌شود تا اندازهٔ فریم به حداقل اندازهٔ مجاز برساند.

دلیل دیگر (و بسیار مهمتر) در محدودیت حداقل اندازه برای هر فریم، آنست که مبادا ایستگاه قبل از رسیدن اوّلین بیت از فریمش به انتهای کابل، ارسال خود را به پایان برساند زیرا ممکن است تصادم بوجود بیاید [و چون ارسال فریم خاتمه یافته ، ایستگاه از این موضوع باخبر نخواهد شد. -م] این مشکل در شکل ۴-۱۸ نشان داده شده است. در زمان t=0 ایستگاه A در یک طرف شبکه شروع به ارسال فریم روی کابل می‌کند. فرض کنید زمان تاخیر رسیدن فریم به انتهای کابل τ ثانیه باشد. دقیقاً قبل از رسیدن فریم به انتهای کابل (یعنی در لحظهٔ $\tau - \varepsilon$) آخرین ایستگاه متصل به کابل یعنی B شروع به ارسال فریم خود می‌نماید. حال وقتی B تشخیص می‌دهد که توان دریافتی از کانال از توان ارسالی بیشتر است متوجه بروز تصادم می‌شود و با قطع ارسال خود و تولید یک نویز ۴۸ بیتی قوی، بروز تصادم را به اطلاع بقیهٔ ایستگاهها می‌رساند. بعبارت دیگر با ارسال یک نویز قوی این اطمینان حاصل می‌شود که هیچ ایستگاهی از پدیدهٔ تصادم بی‌خبر نخواهد ماند. در زمان حدود 2τ، فرستنده متوجه نویز شده و از ارسال دست می‌کشد. سپس قبل از تلاش مجدد برای ارسال، به اندازهٔ یک عدد تصادفی صبر می‌کند.



شکل ۴-۱۸. کشف تصادم می‌تواند تا زمان 2τ طول بکشد.

اگر ایستگاه تلاش کند فریمی کوتاه ارسال نماید و تصادم پدید بیاید، اگر چه ممکن است از تصادم مطلع شود ولی چون قبل از زمان بازگشت نویز (2τ) ، ارسال فریم به پایان رسیده ، فرستنده به غلط نتیجه می‌گیرد که فریم او

با موفقیت ارسال شده است و تکرار ارسال ممکن نخواهد بود. برای اجتناب از چنین وضعیتی ، تمام فریمها بایستی از لحاظ زمانی از (2τ) بیشتر باشند تا اطمینان حاصل شود که قبل از خاتمهٔ ارسال فریم، تصادم کشف خواهد شد. برای شبکهٔ اترنت 10Mbps با کابلی به طول حداکثر ۲۵۰۰ متر و چهار تکرارکننده (طبق تعریف 802.3) ، زمان رفت و برگشت سیگنال (با احتساب تاخیر انتشار چهار تکرارکننده) در بدترین حالت تقریباً پنجاه میکروثانیه محاسبه شده که قطعاً قابل صرفنظر نیست. بدین ترتیب حداقل طول فریم باید به اندازه‌ای باشد که از لحاظ زمانی حداقل ۵۰ میکروثانیه باشد. در سرعت 10Mbps هر بیت ۱۰۰ نانوثانیه طول می‌کشد لذا کوچکترین طول فریم که عملکرد صحیح شبکه را تضمین می‌کند ۵۰۰ بیت خواهد بود. برای افزودن به حاشیه امنیت این مقدار ، ۵۱۲ بیت معادل ۶۴ بایت در نظر گرفته شده است. در فیلد Pad از فریمهای کوچکتر از ۶۴ بایت ، باید آنقدر دادهٔ زائد اضافه شود تا طول آن به ۶۴ بایت برسد.

به تناسب بالا رفتن سرعت شبکه یا باید طول کانال را کاهش داد یا آنکه به طول حداقل فریمها افزود. در یک شبکهٔ LAN با کابلی به طول ۲۵۰۰ متر و نرخ ارسال 1 Gbps ، طول حداقل هر فریم، ۶۴۰۰ بایت خواهد بود! در عوض اگر بیشترین فاصلهٔ دو ایستگاه به ۲۵۰ متر کاهش یابد طول حداقل فریم به ۶۴۰ بایت کاهش خواهد یافت. این محدودیت زمانی که به سمت شبکه‌های چند گیگابیتی حرکت کنیم بسیار مشکل آفرین خواهد بود.

آخرین فیلد از فریم اترنت فیلد Checksum (کد کشف خطا) است. در حقیقت این فیلد یک کد ۳۲ بیتی استخراج شده از داده‌هاست. هرگاه تعدادی از بیتهای فریم (در اثر نویز روی کانال) اشتباه دریافت شوند، کد Checksum آن نادرست خواهد بود و بدین ترتیب خطا کشف می‌شود. کد کشف خطای به کار رفته در اترنت از نوع CRC است که آنرا در فصل سوم تشریح کردیم. این کدها فقط برای کشف خطا کاربرد دارند و قادر به تصحیح خطا نیستند.

زمانی که IEEE اترنت را استاندارد کرد، دو تغییر کوچک در قالب فریم DIX ایجاد کرد؛ این تغییرات در شکل ۴-۱۷-ب مشهود است. اولین تغییر آن بود که طول Preamble (دیباچه) را به ۷ بایت کاهش داد و بایت هشتم را به عنوان «بایت مشخص کنندهٔ ابتدای فریم» (Start Of Frame Delimiter) به کار برد تا با استانداردهای 802.4 و 802.5 سازگار باشد. تغییر دوّم آن بود که فیلد Type را به فیلد Length تغییر کاربری داد. البته در اینصورت گیرنده نمی‌تواند تشخیص بدهد که با فریم دریافتی چه کند ولی این مسئله با اضافه کردن یک سرآیند اضافی به قسمت داده حلّ شده است. قالب قسمت داده را در همین فصل و در توضیح «کنترل منطقی لینک» تشریح خواهیم کرد. متاسفانه در زمان ارائه 802.3 سخت‌افزار و نرم‌افزار زیادی بر اساس اترنت DIX در حال استفاده بود و شرکتهای سازنده و کاربران، چندان علاقمند به تبدیل فیلد Type به Length نبودند! در سال ۱۹۹۷، IEEE کوتاه آمد و اذعان کرد که هر دو روش قابل قبول است. خوشبختانه، تمام مقادیری که در فیلد Type درج می‌شد بالاتر از عدد ۱۵۰۰ بودند. بدین ترتیب قرار بر آن شد که اگر عدد درون این فیلد از ۱۵۰۰ بیشتر باشد به عنوان فیلد Type فرض شود و در غیر اینصورت بعنوان فیلد Length تعبیر گردد. حال دیگر IEEE می‌توانست همه را راضی نگه دارد: چه آنهائی که از استاندارد او استفاده می‌کردند و چه آنهائی که می‌خواستند کار با استاندارد DIX را بدون احساس کمبود ادامه بدهند!!

۴-۳-۴ الگوریتم عقب‌گرد نمایی

در این بخش بررسی خواهیم کرد که وقتی تصادم به وقوع می‌پیوندد روال تولید عدد تصادفی [به منظور انتظار و تلاش مجدد] چگونه است. الگوی ما کماکان مدل شکل ۴-۵ می‌باشد. پس از بروز تصادم ، زمان به تعدادی برش مجزا تقسیم می‌شود؛ طول این برشها معادل با بیشترین زمان رفت و برگشت سیگنال برروی کانال (یعنی 2τ) است. با در نظر گرفتن بیشترین طول کابل در اترنت ، هر یک از این برشهای زمانی معادل ۵۱۲ بیت یعنی ۵۱/۲

میکروثانیه خواهد بود.

پس از اولین تصادم هر ایستگاه قبل از تلاش مجدد، به صورت تصادفی صفر یا یک برش زمانی منتظر می‌ماند.(یعنی بصورت تصادفی یکی از اعداد ۰ یا ۱ را تولید می‌کند.) اگر دو ایستگاه با هم تصادم کنند و اعداد تصادفی تولید شده مشابه باشند، تصادم تکرار خواهد شد. لذا در دومین تصادم متوالی، یکی از اعداد ۰ ، ۱ ، ۲ ، ۳ انتخاب و بهمان تعداد برش زمانی انتظار می‌کشد. [یعنی اگر عدد ۲ باشد ۵۱/۲ × ۲ میکروثانیه منتظر می‌ماند.] اگر سومین تصادم متوالی رخ بدهد (که احتمال چنین رخدادی ۰/۲۵ است) تعداد برشهای زمانی انتظار، عددی تصادفی بین صفر تا ۱ - 2^3 (۷~۰) خواهد بود.

بطور عامّ در تصادم پیاپی iام، عددی تصادفی بین صفر تا ۱ - 2^i انتخاب می‌شود و متناسب با عدد انتخابی، بر مبنای برشهای ۵۱/۲میکروثانیه‌ای منتظر می‌ماند. با این حال پس از دهمین تصادم متوالی،بازه تولید اعداد تصادفی بین صفر تا ۱۰۲۴ ثابت خواهد ماند. پس از شانزدهمین تصادم پیاپی، کنترلر کارت شبکه، دیگر ادامه نداده و پیغامی مبنی بر وجود اشکال جدّی در شبکه، به کامپیوتر گزارش خواهد کرد. تشخیص بیشتر، بر عهدهٔ لایه‌های بالاتر است.

این الگوریتم که اصطلاحاً «**عقب‌گرد نمایی دودوئی**» (Binary Exponential Backoff) نامیده می‌شود، بدین دلیل انتخاب شده تا بتواند خود را بصورت پویا با هر تعداد ایستگاه که در تلاش برای ارسال هستند، تطبیق بدهد. اگر در هر تصادم، زمان تصادفی انتظار به صورت ثابت و در محدودهٔ صفر تا ۱۰۲۳ انتخاب می‌شد اگر چه احتمال دو تصادم متوالی بسیار ناچیز بود ولی در عوض زمانی که ایستگاه‌های تصادم‌کننده باید صبر می‌کردند، بالغ بر صدها بازهٔ زمانی می‌شد و تاخیر بالا می‌رفت. برعکس، اگر بطور دائم یکی از اعداد صفر یا یک انتخاب شود آنگاه اگر به فرض صد ایستگاه سعی در ارسال کنند تصادم‌های پیاپی آنقدر تکرار می‌شود تا زمانی که ۹۹ ایستگاه تصادفاً ۱ و یکی از آنها ۰ را انتخاب کند. این اتفاق ممکن است سالها طول بکشد! اگر الگوریتمی داشته باشیم که در آن بازه‌های زمانی انتظار در پی تصادمات متوالی، بطور نمائی رشد کند، این اطمینان حاصل می‌شود وقتی تعداد ایستگاه‌های آمادهٔ ارسال کم است تاخیر کمی بوجود بیاید و وقتی تعداد ایستگاه‌های آماده ارسال زیاد است در مدت زمان معقولی مسئلهٔ تصادم حلّ شود. گذاشتن سقف ۱۰۲۳ ، اجازهٔ رشد بیش از اندازهٔ زمان انتظار را نخواهد داد.

بگونه‌ای که قبلاً اشاره کردیم در CSMA/CD دریافت فریم تایید نمی‌شود [یعنی پس از ارسال فریم سالم، دریافت آن گزارش نخواهد شد]. از آنجایی که جان به در بردن از تصادم تضمین‌کنندهٔ عدم خرابی بیتها در اثر نویز کانال نیست، لذا در مقصد هر فریم باید بررسی و در صورت صحّت ، پیغام تصدیق (ACK) برگردانده شود. طبعاً پیغام تایید وصول، خودش یک فریم معمولی است که برای ارسال آن نیز همانند فریم داده [برای در اختیار گرفتن کانال] باید مبارزه شود. با این وجود با یک تغییر ساده در الگوریتم رقابت، می‌توان به دریافت پیغام ACK سرعت بخشید.(Tokoro and Tamaru , 1977) تمام کاری که باید انجام شود آنست که پس از خاتمهٔ ارسال فریم، اولین بازهٔ زمانی به ایستگاه مقصد اختصاص داده شود تا پیغام ACK خود را ارسال نماید. [فریم ACK بسیار کوتاه است.] متاسفانه در استاندارد اترنت، چنین قابلیتی گنجانیده نشده است.

۵-۳-٤ کارائی (بازده) اترنت

در اینجا اجازه بدهید کارآئی اترنت را در شرایط بار سنگین و ثابت، یعنی شرایطی که در آن همیشه k ایستگاه آماده ارسال هستند، بررسی نماییم. تحلیل دقیق الگوریتم عقب‌گرد نمایی پیچیده است. به جای آن روش «متکالف» و «باگز» را دنبال کرده و فرض می‌کنیم که احتمال تکرار ارسال فریم در هر برش رقابت، ثابت باشد. اگر هر ایستگاه در خلال برش رقابت با احتمال p اقدام به ارسال فریم نماید احتمال آنکه یک ایستگاه در همان برش موفق به

ارسال شود مساویست با:

$$A = k.p.(1 - p)^{k-1}$$

A زمانی حداکثر خواهد شد که p=1/k باشد؛ با فرض p=1/k ، وقتی k به سمت بی‌نهایت میل می‌کند مقدار A ، 1/e خواهد بود. احتمال آنکه دورهٔ رقابت دقیقاً j مرحلهٔ متوالی ادامه یابد، مساوی با $A.(1-A)^{j-1}$ است فلذا میانگین تعداد دفعات تصادم در هر ارسال طبق رابطهٔ زیر بدست می‌آید:

$$\sum_{j=0}^{\infty} j.A.(1-A)^{j-1} = 1/A$$

از آنجایی که هر برش زمانی حدوداً 2τ طول می‌کشد، میانگین زمان رقابت W = 2τ/A خواهد بود. با فرض مقدار بهینه برای p (یعنی 1/e) میانگین تعداد برشهای رقابت هرگز از 2τ.e بیشتر نخواهد شد [زیرامقدار A در رابطهٔ W=2τ/A حداکثر 1/e است] بدین ترتیب W معادل 2τe یعنی حدود 5.45 خواهد بود.

با فرض آنکه تعداد ایستگاه‌های آماده ارسال ، زیاد باشد و ارسال فریم، P ثانیه به طول بینجامد داریم:

$$\text{کارآئی کانال} = \frac{P}{P + 2\tau/A}$$ رابطهٔ ۴-۶

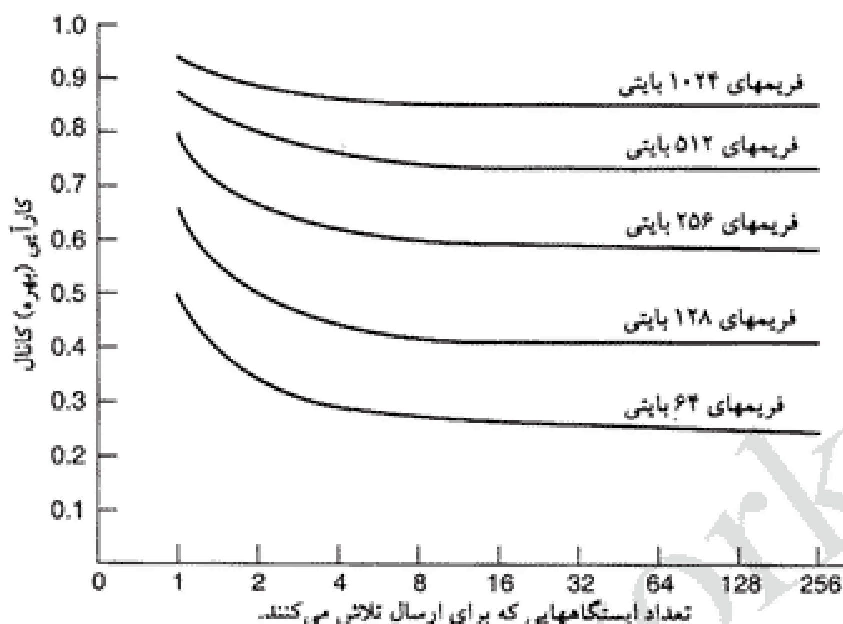
در اینجا می‌بینیم که فاصلهٔ حداکثر کابل بین دو ایستگاه در کارآئی این شبکه وارد می‌شود (زیرا مقدار τ به طول کابل بستگی دارد). این موضوع نشان می‌دهد که برای فواصل طولانی باید به سراغ یک توپولوژی بغیر از آنچه که در شکل ۴-۱۵-الف دیده می‌شود، رفت. هر چه طول کانال زیادتر باشد طول دورهٔ رقابت بیشتر خواهد شد. این نتیجه بروشنی مشخص می‌کند که چرا در استاندارد اترنت، طول کابل محدودیت دارد. بهتر آنست که رابطهٔ ۴-۶ را به صورت واضحتر و بر حسب طول فریم ، F ، پهنای باند شبکه ، B ، طول کابل ، L و سرعت انتشار سیگنال برروی کانال ، C و مقدار بهینهٔ A یعنی 1/e بنویسیم. با قرار دادن P = F/B معادلهٔ ۴-۶ به صورت زیر در می‌آید:

$$\text{کارآئی کانال} = \frac{1}{1 + 2BLe/cF}$$ معادله ۴-۷

وقتی دومین عبارت یعنی (2B.L.e/c.F) بزرگ باشد کارآئی کانال پائین خواهد بود. بالاخص با افزایش پهنای باند یا طول کانال شبکه (یا بطور کلی با افزایش حاصل‌ضرب B.L) ، کارآئی شبکه (برای فریمهای با طول مشخص) کاهش خواهد یافت. متاسفانه بیشتر پژوهشهایی که برای بالا بردن سرعت سخت‌افزار شبکه انجام می‌شود، این حاصل‌ضرب را افزایش می‌دهد زیرا امروزه عموم مردم به پهنای باند بالا و کانالهای طولانی نیاز دارند (مثلاً برای شبکهٔ فیبر نوری MAN) و محاسبهٔ فوق نشان می‌دهد که شبکه اترنت برای چنین محیط‌هائی مناسب نیست. در ادامهٔ همین فصل خواهیم دید که روش دیگر ی برای پیاده‌سازی اترنت پیشنهاد شده که در مبحث اترنت مبتنی بر سوئیچ، بدان خواهیم پرداخت.

در شکل ۴-۱۹، منحنی کارآئی کانال بر حسب تعداد ایستگاه‌های آماده جهت ارسال و با در نظر گرفتن $2\tau = 51.2\mu s$ و نرخ ارسال B=10Mbps ترسیم شده است. [هر یک از منحنی‌ها به ازای یک F یعنی طول فریم مشخص ترسیم شده‌اند.] برای فریمهای ۶۴ بایتی کارآئی کانال چندان جالب نیست. در طرف مقابل، با نظر گرفتن ۶۴ بایت برای هر برش رقابت، میانگین زمان رقابت برای فریمهای ۱۰۲۴ بایتی ، معادل ۱۷۴ بایت بوده و راندمان کانال تقریباً ٪۸۵ است.

برای آنکه تعداد متوسط ایستگاه‌های آمادهٔ ارسال را در بار سنگین محاسبه نمائیم می‌توان از روش زیر بهره گرفت: هر فریم برای مدت زمانی معادل: «طول دورهٔ رقابت به اضافهٔ زمان ارسال فریم» یعنی به اندازهٔ P+w ثانیه، کانال را درگیر خواهد کرد. بدین ترتیب تعداد فریمها در هر ثانیه 1/(P+w) خواهد بود. هرگاه ایستگاه‌ها با نرخ



شکل ۴-۱۹. منحنی کارآئی کانال در اترنت 10 Mbps با فرض برشهای رقابت ۵۱۲بیتی.

میانگین λ فریم برثانیه به تولید فریم مشغول باشند و سیستم «**در حالت** k»[1] باشد، مجموع نرخ تولید فریم در ایستگاه‌های فعال $k\lambda$ فریم برثانیه خواهد بود. از آنجایی که در حالت «**آرامش**» (Equilibrium) بایستی نرخ ورودی و خروجی یکسان باشد لذا می‌توانیم این دو عبارت را معادل هم قرار داده و آنرا بر حسب k حلّ کنیم. (دقت کنید که w خود تابعی از k است.) روش تحلیل پیچیده و دقیقتری توسط (Bertsekas and Gallager, 1992) ارائه شده است.

اشاره به این نکته خالی از لطف نیست که تاکنون حجم بسیار گسترده‌ای از مطالعات در خصوص تحلیل کارآئی شبکهٔ اترنت (و شبکه‌های دیگر) انجام شده است؛ در تمام آنها مجازاً فرض شده که ترافیک ایستگاه‌ها مبتنی بر تابع توزیع پواسون تولید می‌شود. وقتی پژوهشگران مطالعات خود را به مدل حقیقی تولید ترافیک، (در عمل) معطوف کردند، مشخص شد که ترافیک شبکه به ندرت پواسون است و «ترافیک هر ایستگاه خاصّ فقط شبیه به خودش است»! (Paxon and Floyd,1994; and Willinger et al.,1995) این بدین معناست که میانگین‌گیری در یکدورهٔ طولانی از زمان، «میانگین هموارشده» (Smoothed Average) ترافیک را محاسبه نخواهد کرد؛ یعنی متوسط تعداد فریمها در «هر دقیقه از ساعت» دارای واریانس (پراکندگی) بیشتری نسبت به متوسط تعداد فریمها، در «یک ثانیه از هر دقیقه» خواهد داشت. این نتیجه‌گیری بدین معناست که اغلب مدلهای عرضه شده برای ترافیک شبکه قابل اعمال در محیطهای واقعی نیست.

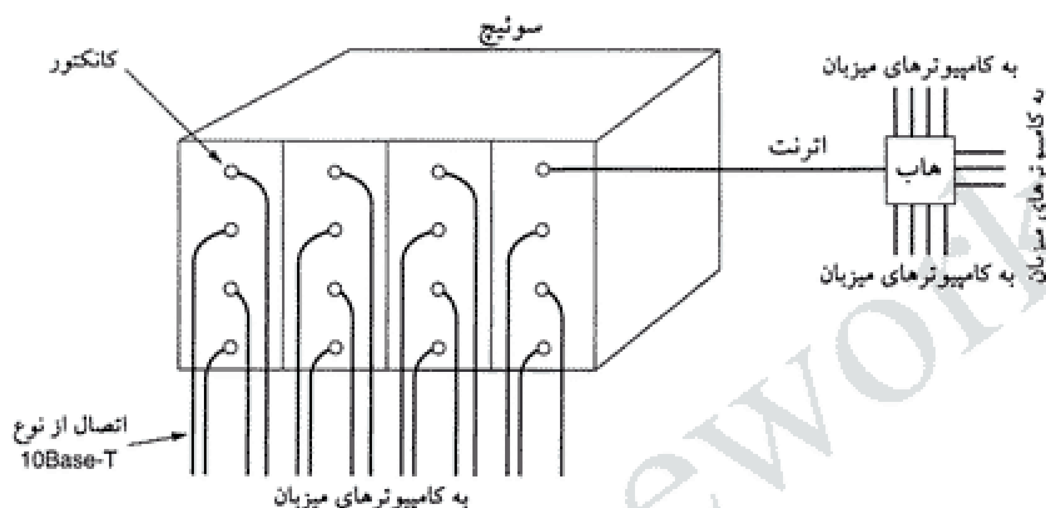
۴-۳-۶ اترنت مبتنی بر سوئیچ

هر چه بر تعداد ایستگاه‌های شبکهٔ اترنت افزوده شود، ترافیک شبکه افزایش یافته و سرانجام شبکهٔ LAN اشباع خواهد شد. یکی از روشهای کاهش این مشکل آنست که از اترنت 10Mbps به سوی اترنت 100Mbps حرکت کنیم ولیکن با رشد کاربردهای چندرسانه‌ای، اترنت 100Mbps یا حتّی 1-Gbps نیز، اشباع خواهد شد.

---

۱. «حالت k» و «حالت آرامش» از اصطلاحات نظریهٔ صف است. برای درک این مبحث باید اندکی در خصوص نظریهٔ صف (Queuing Theory) مطالعه قبلی داشته باشید. -م

خوشبختانه راه دیگری برای مواجهه با بار سنگین وجود دارد: «اترنت مبتنی بر سوئیچ». این مدل در شکل ۴-۲۰ به تصویر کشیده شده است. در مرکز این سیستم «سوئیچ» قرار گرفته که از یک Backplane پرسرعت و فضائی برای نصب ۴ تا ۳۲ «**کارت اتصالی خط**»[1] تشکیل شده است و هر کارت ۱ تا ۸ کانکتور دارد. بطور معمول، کانکتورها از نوع 10Base-T هستند که از طریق سیمهای زوجی بهم‌تابیده، مستقیماً به یک کامپیوتر میزبان متصل می‌شوند.



شکل ۴-۲۰. مثالی ساده از اترنت مبتنی بر سوئیچ.

هرگاه ایستگاهی بخواهد یک فریم اترنت را بفرستد، آن فریم را در قالب عادّی و استاندارد سازماندهی کرده و آنرا برای سوئیچ می‌فرستد. کارت واسط دریافت‌کنندهٔ فریم، ابتدا آدرسهای آنرا بررسی می‌کند تا ببیند آیا گیرندهٔ فریم (مقصد) به همان کارت متصل است؛ اگر مقصد به همان کارت واسط متصل نبود، فریم برروی Backplane پرسرعت ارسال می‌شود تا کارت واسط مقصد آنرا دریافت کند. Backplane با سرعتی حدود چندین گیگابیت در ثانیه و با پروتکلی اختصاصی کار می‌کند.

حال ببینیم وقتی دو ایستگاه متصل به یک کارت مشابه ، به بطور همزمان فریمی را ارسال کنند چه اتفاقی می‌افتد؟ این مسئله به ساختار طراحی آن کارت بستگی دارد. امکان دارد که تمام پورتهای یک کارت از طریق یک سیم، مستقیماً به هم وصل شده و یک شبکهٔ محلی [باس] تشکیل شده باشد. آنگاه، تصادماتی که ممکن است برروی این LAN داخلی رخ بدهد بروش معمولی CSMA/CD حلّ و فصل شده و ارسال مجدّد نیز طبق الگوریتم «عقب‌گرد دودوئی نمایی» (Binary Exponential Backoff) انجام می‌گیرد. در چنین کارتهائی، از بین ایستگاه‌های متصل به هر کارت، در آن واحد فقط یک ایستگاه می‌تواند ارسال کند در حالیکه ایستگاههای متصل به کارتهای متفاوت می‌توانند بطور موازی [همزمان] ارسال داشته باشند. در این طرح هر کارت برای خودش یک «**حوزهٔ تصادم**»[2] (Collision Domain) تشکیل داده و این حوزه، مستقل از حوزهٔ تصادم دیگر کارتهاست. هرگاه فقط یک ایستگاه در هر حوزه وجود داشته باشد [یعنی فقط یک ایستگاه در هر کارت] آنگاه تصادم منتفی است و کارآئی افزایش خواهد داشت.

در انواع دیگر کارتها، هر یک از پورتهای ورودی دارای بافری اختصاصی هستند و بدین ترتیب فریمهای

---

۱. Plug-in Line Card

۲. ایستگاههایی که مستقیماً به یک کانال مشترک متصلند و برای ارسال بروش CSMA/CD رقابت می‌کنند اصطلاحاً در یک «حوزه تصادم» قرار دارند. -م

ورودی در درون حافظهٔ RAM کارت، ذخیره می‌شوند. در این ساختار تمام ایستگاه‌ها می‌توانند بطور همزمان و دوطرفه (Full Duplex)، ارسال و دریافت داشته باشند؛ عملی که در ساختار معمولی CSMA/CD ، با یک کانال واحد ممکن نیست. هنگامی که یک فریم بطور کامل دریافت شود، کارت مربوطه می‌تواند بررسی کند که آیا ماشین مقصد به پورتی برروی همان کارت متصل است یا باید به سمت پورتی بر روی کارت دیگر هدایت شود. در حالت اوّل، فریم مستقیماً به سمت مقصد ارسال می‌شود. در حالت دوّم، فریم از طریق Backplane به سمت کارت مناسب هدایت می‌گردد. در این طراحی، هر پورت برای خود «حوزهٔ تصادم» کاملاً مستقل دارد، یعنی هرگز تصادم رخ نمی‌دهد. در این ساختار، ظرفیت کلّ سیستم در مقایسه با 10Base5 (که در آن برای کل سیستم یک حوزهٔ تصادم واحد وجود دارد) چندین برابر افزایش خواهد داشت.

از آنجایی که سوئیچ، فریمهای استاندارد اترنت را از پورتهای ورودی خود می‌پذیرد لذا می‌توان از برخی پورتها در نقش «متمرکز کننده» (Concentrator) استفاده کرد. در شکل ۴-۲۰ پورت سمت چپ/بالایی سوئیچ، بطور مستقیم به یک ایستگاه واحد متصل نیست بلکه مثلاً به یک هاب ۱۲ پورت متصل است. وقتی فریمی به هاب وارد می‌شود، بروش معمول برای بدست آوردن کانال رقابت می‌کند؛ در حالیکه ممکن است تصادم رخ داده و از الگوریتم عقب‌گرد نمایی استفاده شود. فریمی که موفق به ارسال روی کانال شده و به سوئیچ وارد گردد، همانند یک فریم معمولی با آن برخورد می‌شود یعنی برای رسیدن به خط خروجی مناسب از طریق Backplane ارسال و به مقصد هدایت خواهد شد.

اگر چه هابها ارزانتر از سوئیچها هستند ولیکن با سقوط قیمت سوئیچها، هاب به سرعت از صحنه خارج می‌شود، ولی هنوز میراث هابها باقی است!

٤-۳-۷ اترنت سریع

در بدو آن ایّام، سرعت 10Mbps برای اترنت بسیار شگفت‌انگیز و رویایی به نظر می‌رسید؛ همانگونه که ظهور مودم‌های ۱۲۰۰bps در دورانی که فقط مودم‌های آکوستیکی و ابتدائی ۳۰۰bps رایج بود، پدیده‌ای اعجاب‌انگیز محسوب می‌شد؛ ولیکن هر پدیدهٔ نوظهوری به سرعت فرسوده و محو می‌شود. به عنوان نتیجهٔ قانون پارکنسیون می‌توان منتظر زمانی بود که ارسال داده‌ها با نرخی معادل با پهنای باند طبیعی هر کانال ارسال شوند. (قانون پارکینسون بیان می‌کند که: «یک کار تا زمانی که وقت برای تکمیل شدن داشته باشد، ادامه می‌یابد.») برای افزایش سرعت شبکه‌ها، گروه‌های صنعتی مختلف دو طرح جدید مبتنی برشبکهٔ حلقه (Ring) و فیبرنوری ارائه کردند:یکی از آنها طرح شبکهٔ FDDI[1] و دیگری طرح Fibre Channel[2] نام داشت. برای کوتاه کردن داستان مفصل آنها، باید اشاره کنیم که اگر چه از هر دو به عنوان ستون‌فقرات شبکه استفاده شد ولیکن هیچیک از آنها نتوانست برای اتصال کامپیوترهای رومیزی به کاربرود، چراکه در این دو شبکه، مدیرت ایستگاه‌ها بسیار پیچیده بود و نیاز به تراشه‌های پیشرفته داشت که به گران شدن قیمت آنها می‌انجامید؛ سرانجام پروندهٔ آنها به تاریخ پیوست. درسی که باید از سرنوشت اینگونه شبکه‌ها گرفت آنست که باید هر سیستمی را ساده و سهل طراحی کرد.

بهر تقدیر، ناکامی و شکست شبکه‌های LAN مبتنی بر فیبرنوری [مثل FDDI]، کمبود و خلاء اترنت سریعتر از 10Mbps را هر چه بیشتر نمایان می‌کرد. در بسیاری از محیطها، افراد به پهنای باند بیشتری نیاز داشتند و برای رفع این مشکل به ناچار چندین شبکهٔ محلی 10Mbps را نصب و از طریق تکرارکننده (Repeater) ، پل

---

۱. Fiber Distributed Data Interfce

۲. نام این طرح Fibre Channel است نه Fiber Channel، زیرا ویرایشگر این طرح بریتانیایی بوده نه آمریکایی! -اندی تاننبام

(Bridge) یا مسیریاب (دروازه ـ Gateway ـ) به هم متصل کرده بودند. این ساختار بهم‌پیچیده، گاهی چنان سردرگم و پراشکال می‌شد که مسئول شبکه احساس می‌کرد این شبکه‌ها به وسیلهٔ آدامس بادکنکی یا نخ به هم کوک خورده‌اند!

چنین فضا و شرایطی، IEEE را بر آن داشت تا در سال ۱۹۹۲ کمیتهٔ 802.3 را دور هم جمع کند تا در خصوص شبکهٔ محلی سریعتر از ده‌مگابیت تصمیم بگیرند. یک پیشنهاد آن بود که 802.3 بهمان شکل اصلی حفظ شده و فقط سرعت آن افزایش یابد. پیشنهاد دیگر آن بود که 802.3 بطور کلّ از نو تدوین شده و ویژگی‌های جدیدی مثل ارسال ترافیک بی‌درنگ و مبادله دیجیتالی سیگنال صوت به آن اضافه شود ولی در عین حال (بدلیل مسائل بازار و فروش) نام قبلی آن حفظ شود. پس از بحث و مناقشهٔ فراوان، کمیته تصمیم گرفت که ساختار شبکهٔ 802.3 را به همان شکل اصلی حفظ کند و فقط به سرعت آن بیفزاید.

گروهی که پیشنهاد آنها در خصوص طراحی مجدّد 802.3 پذیرفته نشد برای خودشان کمیته تشکیل دادند و شبکهٔ محلّی پیشنهادی خود را با نامی دیگر استاندارد کردند که نهایتاً شبکهٔ 802.12 پدید آمد. (کاری که معمولاً هر شخصی با دید صنعتی یا اقتصادی در چنین شرایطی انجام خواهد داد انجام طرحی است که درست و منطقی به نظر می‌رسد. ولی بازار چیز دیگری می‌گوید! تاریخ نشان داد که شبکهٔ 802.12 که شرکت هیولت پاکارد برروی آن سرمایه‌گذاری کرد اقبالی بدست نیاورد!)

به سه دلیل زیر، کمیتهٔ 802.3 تصمیم گرفت اترنت را بدون تغییر در بنیان آن، فقط توسعه بدهد:

۱. نیاز به سازگاری شبکهٔ جدید با شبکه‌های اترنت موجود

۲. نگرانی از آنکه پروتکل جدید مشکلات پیش‌بینی‌نشده داشته باشد!

۳. تمایل به آنکه قبل از تغییر تکنولوژی بتوانند کار را به اتمام برسانند و مشمول زمان نشود.

کار به سرعت انجام گرفت (البته با سرعتی که در عرف کمیتهٔ استاندارد است) و حاصل کار، استاندارد 802.3u بود که به صورت رسمی در ژوئن ۱۹۹۵ توسط IEEE تائید و معرفی شد. از دیدگاه فنی، 802.3u استاندارد جدیدی محسوب نمی‌شود بلکه یک ضمیمهٔ مکمل برای استاندارد موجود 802.3 است. (تا برسازگاری با استاندارد قبل تاکید شده باشد). از آنجایی که عموم افراد به استاندارد جدید به جای 802.3u ، «اترنت سریع» (Fast Ethernet) می‌گویند ما نیز همین اصطلاح را به کار خواهیم گرفت.

ایدهٔ اصلی در اترنت سریع بسیار ساده بود: تمام ویژگیها مثل قالب فریم ، واسطها، قواعد و الگوریتمها را بدون تغییر نگاه داشته و فقط «زمان یک بیت» را از 100 نانو ثانیه به 10 نانو ثانیه کاهش بدهیم. از دیدگاه فنی، این کار در شبکه اترنت با کابل 10Base5 یا 10Base2 بسادگی میسر است و به شرط آنکه طول حداکثر کانال با ضریب ده کاهش یابد ، تصادمها نیز بموقع کشف خواهد شد. با این وجود محاسن سیم‌کشی مبتنی بر 10Base-T (یعنی استفاده از زوج سیم‌های به هم‌تابیده) بقدری مفید و چشمگیر بود که اترنت سریع کلاً بر اساس این نوع کابل طراحی شد. لذا کلّ سیستم اترنت سریع، از هاب یا سوئیچ بهره می‌گیرد و استفاده از کابلهای چندانشعابی ، کانکتورهای BNC یا انشعاب‌های تزریقی (Vampire Tap) مجاز نیست.

ولیکن هنوز چند گزینهٔ دیگر باقی مانده بود که بایستی برای آنها نیز تصمیم گرفته می‌شد؛ مهمترین آنها، انتخاب انواع سیم زوجی بود که اترنت سریع باید از آنها پشتیبانی می‌کرد. یکی از نامزدها کابل زوجی ردهٔ ۳ (Cat 3) بود. [کابل Cat 3 برای سیم‌کشی تلفن در ساختمانها به کار می‌رود.] استدلال این انتخاب آن بود که در دنیای غرب در هر دفتر از ساختمانهای اداری حداقل چهار جفت سیم Cat 3 (یا حتی بهتر از Cat 3) از قبل وجود دارد که در فاصله‌ای حدود صد متر به جعبهٔ تقسیم تلفن کشیده شده است. بنابراین با استفاده از کابل زوجی Cat 3 این امکان وجود داشت که بدون نیاز به سیم‌کشی مجدّد ساختمان، بتوان کامپیوترهای رومیزی را از طریق اترنت

سریع به هم متصل کرد؛ این ویژگی، امتیاز بزرگی برای اغلب سازمانها و موسسات به حساب می‌آمد.
بزرگترین اشکال سیم‌های زوجی Cat 3 آنست که قادر نیستند سیگنالی با نرخ تغییر 200Mbaud/sec را در فاصلهٔ صد متر حمل کنند (100Mbps در روش منچستر معادل 200Mbaud/sec است)؛ در استاندارد 10Base-T ، فاصلهٔ صد متر ، بیشترین فاصلهٔ مجاز کامپیوتر از هاب است. از طرفی کابلهای ردهٔ ۵(یعنی Cat 5) می‌توانند چنین سیگنالی را به راحتی در فاصلهٔ صد متری منتقل کنند؛ فیبر نوری حتی قادر است با نرخ بسیار بالاتر، این کار را انجام بدهد.

تصمیم نهائی بر آن شد که مطابق با مشخصات شکل ۴-۲۱ ، هر سه گزینه در استاندارد جدید مجاز شمرده شود، ولیکن به گزینهٔ استفاده از سیم‌های Cat 3 بهبودهائی داده شد تا بتواند سیگنال با نرخ مورد نیاز را حمل کند.

نام کابل	نوع کابل	حداکثر طول هر قطعه	مزایا
100Base-T4	Twisted pair	100 m	از کابلهای معمولی تلفن (UTP Cat 3) استفاده می کند.
100Base-TX	Twisted pair	100 m	ارسال دوطرفه کامل با نرخ 100Mbps و کابل UTP Cat 5
100Base-FX	Fiber optics	2000 m	ارسال دوطرفه کامل با نرخ 100Mbps در فواصل طولانی

شکل ۴ـ۲۱. کابل‌کشی اترنت سریع.

در الگوی سیم‌کشی با کابلهای معمولی Cat 3 (از نوع UTP) که 100Base-T4 نامیده می‌شود از نرخ سیگنالینگ ۲۵ مگاهرتز استفاده شده است فلذا فقط ۲۵ درصد سریعتر از استاندارد ۲۰ مگاهرتز در اترنت استاندارد است (با مرور شکل ۴-۱۶ به یاد بیاورید که درکدینگ منچستر، هر یک از ده میلیون بیت در ثانیه نیاز به دو پالس ساعت دارد.) بهر حال برای تامین پهنای باند مورد نیاز در 100Base-T4 به چهار زوج سیم Cat 3 نیاز است. [۴×۲۵MHz] از آنجایی که سالهاست در استاندارد سیم‌کشی تلفن ساختمانها، از کابلهایی استفاده می‌شود که دارای چهار جفت سیم هستند لذا بسیاری از ادارات چنین امکانی را در اختیار داشتند. البته این موضوع بدین معناست که مرکز تلفن را کنار بگذارید ولی در عوض هزینهٔ بسیار کمتری برای داشتن سیستم پست الکترونیکی سریعتر صرف خواهد شد!

از بین چهار زوج سیم، از یکی به عنوان خط ورودی دائم به هاب، از یکی به عنوان خط خروجی دائم از هاب و از دوتای دیگر بصورت قابل تنظیم در جهت فعلی ارسال [از هاب به کامپیوتر یا بالعکس] استفاده می‌شود. برای صرفه‌جویی در پهنای باند مورد نیاز، از روش کدینگ منچستر استفاده نشده است چراکه با ابداع مولدهای مدرن سیگنال ساعت و در فاصله‌ای بدین کوتاهی، اصولاً نیازی هم به کدینگ منچستر نیست. بجای آن از سیگنالهای Ternary استفاده شده که در این کدینگ، در هر سیکل سیگنال ساعت یکی از مقادیر ۰ ، ۱ یا ۲ ارسال می‌شود.(یعنی پالسها سه‌سطحی هستند.) با داشتن سه زوج سیم در جهت ارسال و روش سیگنالینگ سه‌سطحی ، ۲۷ (۳×۳×۳) سمبل مختلف قابل‌ارسال است؛ بدین ترتیب با ارسال هر سمبل، می‌توان ۴ بیت را (به همراه مقداری افزونگی) انتقال داد. [۴ بیت معادل ۱۶ سمبل و ۱۱ حالت افزونگی] ارسال چهار بیت در هر سیکل از سیگنالهای ۲۵ مگاهرتزی، نرخی معادل 100 مگابیت برثانیه را فراهم می‌آورد. همچنین، در جفت‌سیم باقیمانده (زوج چهارم) یک کانال معکوس ۳۳/۳ مگابیت برثانیه ایجاد می‌شود. این ساختار که اصطلاحاً 8B/6T نامیده می‌شود (بمعنای آنکه هشت بیت در چهار تریت ـTritـ نگاشته می‌شود) روشی جالب و جذابی نیست ولی بهرحال با ساختار موجود سیم‌کشی، کار می‌کند.

الگوی سیم‌کشی با سیم‌های زوجی Cat 5 (که اصطلاحاً 100Base-TX نامیده می‌شود) ساده‌تر است زیرا این سیمها قادر به حمل سیگنالهایی با نرخ ۱۲۵ مگاهرتز هستند. بدین ترتیب برای هر ایستگاه فقط به دو زوج سیم

نیاز است: یکی ورودی به هاب و دیگری خروجی از هاب. در این الگو نیز از روش کدینگ معمولی استفاده نشده است: در عوض از روش 4B/5B بهره گرفته شده که مشابه و سازگار با FDDI است. در روش 4B/5B ، هر گروه متشکل از پنج کلاک متوالی (معادل پنج بیت) و شامل ۳۲ حالت مختلف است. ۱۶ تا از این ترکیبات برای ارسال ۴ بیت داده به کار می‌رود. [عبارت ساده‌تر در روش 4B/5B به ازای هر چهار بیت، پنج بیت ارسال می‌شود.] برخی از ۱۶ حالت باقیمانده برای عملیات کنترلی نظیر مشخص کردن ابتدا و انتهای فریم کاربرد دارد. ترکیبات شانزده‌گانهٔ داده به نحوی انتخاب شده‌اند که در الگوی سیگنال تولیدی، لبهٔ لازم برای سنکرون ماندن سیگنال ساعت وجود داشته باشد. سیستم 100Base-TX دو طرفهٔ همزمان است: ایستگاه می‌تواند داده‌ها را با نرخ ۱۰۰ مگابیت در ثانیه ارسال و بطور همزمان دریافت نماید. اغلب به روشهای 100Base-TX و 100Base-T4 به اختصار 100Base-T گفته می‌شود.

آخرین گزینه، 100Base-FX است که در آن از دو رشتهٔ فیبرنوری مالتی‌مود (Multimode) استفاده می‌شود؛ هر یک از تارهای نوری قادر به حمل ۱۰۰ مگابیت در ثانیه به صورت همزمان هستند. مضاف بر این، فاصلهٔ بین ایستگاه و هاب تا ۲ کیلومتر قابل افزایش است.

در پاسخ به نیاز عمومی، کمیتهٔ IEEE 802 در سال ۱۹۹۷ روش کابل‌کشی جدیدی به نام 100Base-T2 به استاندارد اضافه کرد که اجازه می‌داد اترنت سریع با دو زوج سیم معمولی Cat 3 کار کند ولیکن بدلیل استفاده از روش کدینگ خاصّ ، به یک پردازندهٔ سیگنال دیجیتال (DSP Processor) پیچیده نیاز است که این انتخاب را اندکی گران قیمت می‌کند. اکنون از این ساختار به دلائلی مثل پیچیدگی و قیمت بالا و این حقیقت که بسیاری از ساختمان‌های اداری به سیم‌کشی با کابل Cat 5 تن داده‌اند ، به ندرت استفاده می‌شود.

در 100Base-T ، جهت اتصال ایستگاه‌ها به یکدیگر می‌توان از دو نوع ابزار استفاده کرد: هاب یا سوئیچ. (شکل ۴-۲۰) در هاب تمام خطوط ورودی (یا حداقل تمام ورودی‌هائی که به یک کارت واحد وارد می‌شوند) به صورت منطقی به هم متصل هستند و یک حوزهٔ تصادم واحد را ایجاد می‌کنند. در این ابزار دقیقاً مثل اترنت، تمام قواعد استاندارد (نظیر الگوریتم عقب‌گرد نمایی) اعمال می‌شود، خصوصاً آنکه در هر لحظه تنها یک ایستگاه می‌تواند ارسال داشته باشد؛ بعبارت دیگر، در هاب ماهیت ارتباط ، «دوطرفهٔ غیرهمزمان» (Half Duplex) است.

در یک سوئیچ هر یک از فریمهای ورودی برروی حافظهٔ کارت متصل به خط ، بافر می‌شود و در صورت نیاز از طریق یک Backplane بسیار سریع، از کارت مبداء به کارت مقصد هدایت می‌گردد. ساختار Backplane استانداردسازی نشده و نیازی هم بدین امر نبوده است چرا که به عنوان بخشی پنهان در درون سوئیچ انجام وظیفه می‌کند. با تکیه بر تجارب گذشته می‌توان پیش‌بینی کرد که تولیدکنندگان سوئیچ به شدت در حال رقابت هستند تا Backplane سوئیچها سریعتر شده و کارآئی کلّ سیستم افزایش یابد. از آنجایی که کابلهای 100Base-FX برای کشف بموقع تصادم در شبکهٔ اترنت بیش از اندازه طولانی هستند، فلذا این کابلها لزوماً باید به سوئیچها متصل شوند؛ بدین ترتیب هر ایستگاه برای خود یک «حوزهٔ تصادم» مستقل پدید آورده و تصادم منتفی خواهد بود. استفاده از هاب در 100Base-FX مجاز نیست.

به عنوان آخرین نکته باید اشاره کرد که تمام سوئیچها (مجازاً) می‌توانند با تلفیقی از ایستگاه‌های 10Mbps و 100Mbps کار کنند تا ارتقاء سیستمهای موجود به سیستمهای سریعتر ساده‌تر شود. در صورت افزایش نیاز یک سایت به تعداد بیشتری ایستگاه 100Mbps ، تنها کافی است به تعداد لازم کارت‌های ورودی خریداری شده و درون سوئیچ نصب شود. [به شکل ۴-۲۰ نگاه کنید.] در حقیقت در خود استاندارد روشی پیش‌بینی شده تا دو ایستگاه بتوانند بر سر نرخ ارسال (10Mbps یا 100Mbps) با هم توافق کرده و ماهیت ارتباط (Full Duplex یا Half Duplex بودن ارتباط) را انتخاب نمایند. بسیاری از محصولات اترنت از این ویژگی برخوردار هستند تا

بتوانند به صورت خودکار خودشان را پیکربندی نمایند.

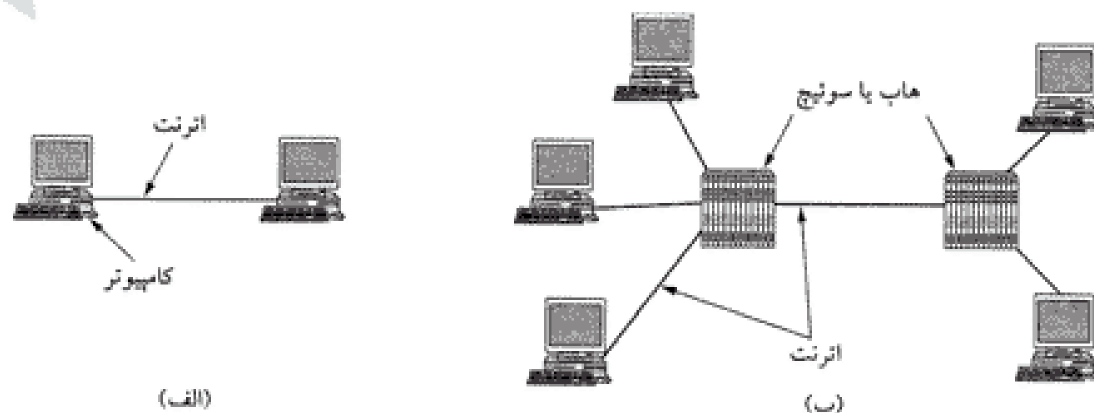
۸-۳-٤ اترنت گیگابیت (Gigabit Ethernet)

هنوز جوهر مستندات استاندارد اترنت سریع، خشک نشده بود که کمیتهٔ 802 کار را برروی اترنت سریعتر از آن آغاز کرد (۱۹۹۵). این تحقیقات سریعاً تحت عنوان اترنت گیگابیت به ثمر رسید و در سال ۱۹۹۸ توسط IEEE با نام 802.3z تصویب شد. پیشنهاد حرف z در نام این استاندارد بدین مضمون بود که اترنت گیگابیت آخرین حلقه در مسیر تکاملی اترنت خواهد بود مگر آنکه کسی بتواند حرف جدیدی پس از z ابداع کند!! در ادامه برخی از ویژگی‌های اساسی اترنت گیگابیت را تشریح خواهیم کرد. برای آگاهی بیشتر می‌توان به مقالهٔ (Seifert, 1998) مراجعه کرد.

اهداف کمیتهٔ 802.3z مشابه با اهداف کمیتهٔ 802.3u بود: اترنت ده برابر سریعتر شود و همچنان با استاندارد موجود اترنت سازگاری داشته باشد. بویژه اترنت گیگابیت باید از خدمات «انتقال دیتاگرام بدون تصدیق دریافت فریم»[1] به صورت تک‌پخشی و چندپخشی[2] و با استفاده از همان ساختار ۴۸ بیتی آدرس موجود، پشتیبانی می‌کرد. همچنین قالب فریمها و طول حداقل و حداکثر هر فریم باید مشابه با قبل انتخاب می‌شد. استاندارد نهائی، تمام این اهداف را برآورده کرده است.

کلّ پیکربندی شبکهٔ اترنت گیگابیت، به جای آنکه همانند اترنت کلاسیک دارای ساختار باس چنداتصالی (Multidrop) باشد صرفاً نقطه‌به‌نقطه (Point To Point) است. [یعنی به جای آنکه همهٔ ایستگاه‌ها به یک کابل مشترک متصل باشند بطور مستقیم به یک سوئیچ متصل می‌شوند. -م] ساده‌ترین الگوی پیکربندی اترنت گیگابیت، در شکل ۴-۲۲-الف نشان داده است؛ در این شکل دو کامپیوتر به صورت مستقیم به هم متصل شده‌اند. الگوی رایج دیگر که در شکل ۴-۲۲-ب دیده می‌شود شامل یک سوئیچ یا هاب است که به چندین کامپیوتر یا چند سوئیچ یا هاب دیگر متصل هستند. در هر دو الگو، به هر کابل اترنت صرفاً دو ابزار متصل است، نه کمتر نه بیشتر [یعنی دو کارت اترنت گیگابیت در دو سر کابل].

اترنت گیگابیت می‌تواند به دو روش عمل کند: «حالت دوطرفهٔ همزمان» (Full Duplex) و «حالت دوطرفهٔ غیرهمزمان» (Half Duplex). عملکرد طبیعی و پیش‌فرض سیستم، حالت دو طرفهٔ همزمان است و بدین ترتیب این امکان وجود دارد که ترافیک داده‌ها بتواند به صورت همزمان در دو جهت جریان داشته باشد. از این حالت زمانی استفاده می‌شود که یک سوئیچ مرکزی وجود داشته و به کامپیوتر دیگر (یا سوئیچهای دیگر) در پیرامون



شکل ۴-۲۲. (الف) اترنت گیگابیت با دو ایستگاه. (ب) اترنت گیگابیت با چند ایستگاه.

---

۱. Unacknowledged Datagram
۲. Unicast & Multicast

خود متصل باشد. در این پیکربندی تمام خطوط بافر شده‌اند[1] و طبعاً هر کامپیوتر یا سوئیچ اجازه دارد هر زمان که‌تمایل داشت فریم خود را ارسال کند. فرستند مجبور نیست کانال را شنود کند و حضور دیگران برروی کانال راتشخیص بدهد چرا که در این ساختار اصولاً هرگونه رقابتی منتفی است و هرگز تصادم رخ نخواهد داد. از آنجایی که بین سوئیچ و کامپیوتر دو کابل مستقل وجود دارد، ارسال همزمان از کامپیوتر به سوئیچ (حتی اگر سوئیچ در حال ارسال فریم برای همان کامپیوتر باشد)، میسّر است. از آنجایی که هیچ رقابتی مطرح نیست لذا پروتکل CSMA/CD کاربردی ندارد و حداکثر طول کانال فقط بر اساس توان سیگنال ارسالی [و میزان تضعیف آن روی کابل] تعیین می‌شود و ربطی به مدت زمان بازگشت نویز حاصل از تصادم [و تاخیر انتشار کابل] ندارد. سوئیچها آزادند که با نرخ مختلف ارسال و به صورت تطبیقی عمل کنند. در اترنت گیگابیت همانند اترنت سریع، از پیکربندی خودکار حمایت می‌شود.

حالت دیگر عملکرد اترنت گیگابیت، حالت دو طرفه غیر همزمان (Half Duplex) است و زمانی کاربرد دارد که کامپیوترها به جای وصل به سوئیچ به هاب متصل باشند. یک هاب فریمهای ورودی را بافر نخواهد کرد و در عوض به صورت داخلی تمام خطوط را به صورت الکتریکی به هم متصل کرده و همانند اترنت کلاسیک کابلی چنداتصالی را شبیه‌سازی می‌کند. در این حالت وقوع تصادم محتمل بوده و به پروتکل CSMA/CD نیاز خواهد بود. از آنجایی که کوتاه‌ترین فریم مجاز (یعنی فریم ۶۴ بایتی)، در اترنت گیگابیت صدبرابر سریعتر از اترنت کلاسیک ارسال می‌شود لذا حداکثر طول کابل بایستی ۱۰۰ برابر کاهش یابد تا آنکه ویژگی بنیادی مورد نیاز در CSMA/CD یعنی «عدم خاتمهٔ ارسال فریم قبل از بازگشت نویز حاصل از تصادم»، برآورده شود.(یعنی طول کابل باید حداکثر ۲۵ متر باشد.) باکابلی به طول ۲۵۰۰ متر و سرعت 1Gbps، قبل از آنکه یک فریم ۶۴بایتی بتواند حتی یکدهم از مسیر خود را طی کند، فرستندهٔ آن ارسال خود را به پایان رسانده و از تصادم احتمالی مطلع نخواهد شد!

کمیتهٔ 802.3z بدین نتیجه رسید که شعاع ۲۵ متر برای اترنت گیگابیت قابل قبول و مناسب نیست، فلذا دو ویژگی جدید به استاندارد افزود تا شعاع شبکه افزایش یابد. اوّلین ویژگی که **«توسیع حامل»** (Carrier Extension) نامیده شده سخت‌افزار را وادار می‌کند تا آنقدر اطلاعات زائد در انتهای فریم معمولی اضافه کند تا طول فریم حداقل ۵۱۲ بایت شود. از آنجایی که این اطلاعات زائد در سخت‌افزار مبداء اضافه و در سخت‌افزار مقصد حذف می‌شوند لذا نرم‌افزار از انجام چنین عملی بی‌اطلاع خواهد بود و هیچ تغییری در ساختار نرم‌افزار موجود نیاز نخواهد بود. البته طبیعی است که فرستادن فریمی ۵۱۲بایتی برای ارسال ۴۶ بایت دادهٔ خام کاربر، بهرهٔ مفیدی معادل ۹ درصد دارد.

دوّمین ویژگی که اصطلاحاً Frame Bursting نامیده شده اجازه می‌دهد که فرستنده دنباله‌ای از چندین فریم را در یک بار ارسال بفرستد. اگر مجموع دنبالهٔ فریمها کمتر از ۵۱۲بایت شود باز هم سخت‌افزار، داده‌های زائد [به انتهای آخرین فریم] اضافه می‌کند. هرگاه تعداد فریمهای منتظر ارسال کافی باشد این روش بسیار کارآمد و نسبت به روش «توسیع حامل» ارجح‌تر است. این ویژگی جدید، شعاع شبکه را به ۲۰۰ متر افزایش می‌دهد که برای بسیاری ادارات و موسسات کافی است. اگرچه در اترنت گیگابیت از CSMA/CD پشتیبانی می‌شود ولیکن تصور آن بسیار دشوار است که سازمانی کارتهای اترنت گیگابیت خریداری و نصب کند ولی آنها را به یک هاب متصل و اترنت معمولی را شبیه‌سازی نماید! [بهرهٔ واقعی اترنت گیگابیت با سوئیچها بدست می‌آید نه با هاب]

اگر چه هابها از سوئیچها بسیار ارزانتر هستند ولی هنوز قیمت کارت‌های اترنت گیگابیت نسبتاً گران است.

---

۱. یعنی داده‌های ارسالی مستقیماً به درون حافظه موقت منتقل می‌شوند. -م

خرید هابهای ارزان قیمت [برای کارتهای گرانقیمت گیگابیتی] و کاستن از بهرهٔ واقعی سیستم، کوته‌فکری به نظر می‌رسد! بهرحال ویژگی «سازگاری با قبل» یکی از نگرانیهای صنایع کامپیوتریست و طبعاً نیاز بوده که کمیتهٔ 802.3z آنرا در استاندارد خود لحاظ کند. [ولیکن دلیلی ندارد از ویژگی‌های جدید آن به بهای سازگاری با قبل صرفنظر شود]

اترنت گیگابیت مطابق با فهرست ۴-۲۳ از کابلهای مسی و فیبرهای نوری پشتیبانی می‌کند. تولید سیگنال با نرخی نزدیک به 1-Gbps بدان معناست که منبع مولد نور باید در زمانی زیر یک نانوثانیه خاموش و روشن شود. مولد نور معمولی یعنی LED نمی‌تواند با این سرعت عمل کند و طبعاً به لیزر نیاز است. استفاده از پرتوهای لیزر با طول موج ۰/۸۵ میکرون (طول موج کوتاه) و ۱/۳ میکرون (طول موج بلند) مجاز است. لیزرهای 0.85 میکرونی ارزانتر هستند ولیکن برروی فیبرنوری تک‌مود (Single Mode) کار نمی‌کنند.

نام کابل	نوع کابل	حداکثر طول هر قطعه	مزایا
1000Base-SX	Fiber optics	550 m	از فیبرهای چندموده ۵۰ و ۶۲/۵ میکرون استفاده می کند.
1000Base-LX	Fiber optics	5000 m	با فیبرهای تک‌موده ۱۰ میکرون یا چندموده ۵۰ و ۶۲/۵ میکرون
1000Base-CX	2 Pairs of STP	25 m	از کابلهای زوجی زره‌دار (STP) استفاده می کند.
1000Base-T	4 Pairs of UTP	100 m	از کابلهای استاندارد UTP Cat 5 استفاده می کند.

شکل ۴-۲۳. کابل‌کشی اترنت گیگابیت.

در اترنت گیگابیت ، استفاده از فیبرهای نوری با قطر ۱۰ ، ۵۰ و ۶۲/۵ میکرون مجاز است. مورد اوّل برای فیبرنوری تک‌مود(Single Mode) و دو مورد بعدی برای فیبرهای چندموده هستند. تمام شش ترکیب مختلف [یعنی ترکیبات مختلف دو نوع لیزر و سه قطر] مجاز نیست ولیکن طول حداکثر کابل به ترکیب مورد استفاده وابسته است. اعدادی که در شکل ۴-۲۳ ارائه شده‌اند برای بهترین حالت ممکن هستند. رسیدن به طول کابل ۵۰۰۰ متری فقط با لیزر ۱/۳ میکرون و صرفاً با کابل تک موده به قطر ۱۰ میکرون امکان‌پذیر است ولیکن این انتخاب علیرغم قیمت گران آن بهترین گزینه برای پیاده سازی ستون فقرات شبکه‌های ناحیه‌ای (Campus) محسوب می‌شود.

در الگوی 1000Base-CX ، از کابلهای مسی زره‌دار کوتاه (STP) استفاده شده است. این انتخاب از یک طرف با فیبرنوری با کارآئی بالا و از طرف دیگر با سیم‌های ارزان‌قیمت UTP در رقابت است زیرا نه به ارزانی UTP است و نه به کارآمدی فیبر نوری! لذا احتمالاً از آن استقبالی نخواهد شد.

آخرین انتخاب، استفاده از چهار زوج سیم Cat 5 است که همزمان با یکدیگر کار می‌کنند. بدلیل آنکه حجم زیادی از این نوع کابل از قبل نصب شده لذا می‌توان از آن به عنوان گونهٔ فقیرانه و کم خرج اترنت گیگابیت یاد کرد!

اترنت گیگابیت برای کدینگ سیگنال روی فیبرنوری از روشی جدید استفاده می‌کند. بهره‌گیری از روش منچستر با سرعت 1-Gbps ، نیاز به تغییر در سطح سیگنال با نرخی معادل 2-G baud/sec خواهد داشت که رسیدن به آن بسیار دشوار بوده و نیاز به پهنای باند بسیار زیادی دارد. در عوض از روشی جدید به نام 8B/10B استفاده شده است. در این روش هر بایت هشت بیتی قبل از ارسال برروی فیبرنوری به یک الگوی ده بیتی نگاشته می‌شود؛ به همین دلیل نام آن 8B/10B انتخاب شده است. از آنجایی که کلمهٔ کد ده بیتی خروجی (که به ازای هر بایت ورودی تولید می‌شود) دارای ۱۰۲۴ حالت مختلف است لذا برای هر یک از ۲۵۶ حالت مختلف ورودی می‌توان یک کلمهٔ کد، متناسب با شرایط کانال انتخاب کرد. برای انتخاب کلمهٔ کد، دو قاعدهٔ زیر به کارگرفته می‌شود:

۱. هیچ کلمهٔ کدی نباید بیش از ۴ بیت مشابه و پشت سرهم داشته باشد.

۲. هیچ کلمهٔ کدی نباید جمعاً بیش از ۶ بیت صفر یا ۶ بیت یک داشته باشد.

این معیارها باعث خواهد شد که در جریان سیگنال تولید شده در خروجی بقدر کافی لبه (Transition) وجود داشته باشد تا این اطمینان حاصل شود که گیرنده با فرستنده هماهنگ و سنکرون باقی مانده و تعداد صفرها و یکهای ارسالی بر روی فیبر، حتی‌الامکان مساوی یا نزدیک به هم باشد. مضاف بر این بسیاری از بایت‌های ورودی می‌توانند بیش از یک کلمهٔ کد هم‌ارز داشته باشند. وقتی که کدکننده، در انتخاب کلمهٔ کد آزادی انتخاب داشته باشد می‌تواند کلمهٔ کدی را انتخاب کند که برآیند تعداد صفرها و یکهائی که تا کنون ارسال شده‌اند حتی‌الامکان معادل باشد. تاکیدی که برروی معادل بودن تعداد صفرها و یک‌ها وجود دارد از آن جهت است که مولفهٔ DC سیگنال حتی‌الامکان پائین بوده و در صورت عبور از مبدلها(Transformers) تغییری در شکل سیگنال ایجاد نشود. اگر چه دانشمندان کامپیوتر تمایلی ندارند که خصوصیات یک ترانسفورمر، نوع کدینگ آنها را تعیین کند ولیکن بهرحال نکته‌ای است که باید رعایت شود.

در اترنت گیگابیت برای الگوی سیم‌کشی 1000Base-T ، از روش کدینگ متفاوتی بهره گرفته شده است چراکه ارسال پالسهای داده بر روی سیم مسی در زمان یک نانو ثانیه بسیار دشوار است. در کدینگ جدید از چهار زوج سیم Cat 5 استفاده شده تا چهار سمبل به صورت همزمان و موازی ارسال شوند. هر سمبول با پنج سطح ولتاژ متفاوت کد می‌شود. این الگو اجازه می‌دهد تا یک سمبول بتواند یکی از حالات 00 ، 01 ، 10 ، یا 11 و یک حالت کنترل خاص را کد نماید. بنابراین در هر سیکل سیگنال ساعت، برروی هر زوج سیم ۲ بیت و برروی کل سیمها ۸ بیت ارسال می‌شود. سیگنال ساعت در فرکانس ۱۲۵ مگاهرتز کار می‌کند که اجازهٔ ارسال یک گیگابیت در هر ثانیه را خواهد داد. دلیل آنکه بجای استفاده از چهار سطح ولتاژ از پنج سطح استفاده شده آنست که برخی از ترکیبات آن برای عملیات کنترلی و فریمینگ باقی بماند.

سرعت 1 Gbps بسیار سریع و بالاست: به عنوان مثال هرگاه یک گیرنده، فقط برای یک میلی‌ثانیه به کاری دیگر مشغول باشد و نتواند بافر ورودی یکی از خطوط را خالی کند، در این وقفهٔ یک‌میلی‌ثانیه‌ای، ۱۹۵۳ فریم در بافر جمع خواهد شد!! همچنین اگر کامپیوتری در شبکهٔ اترنت گیگابیت، داده‌های را برای کامپیوتری در شبکهٔ اترنت کلاسیک بفرستد به احتمال بسیار زیاد داده‌ها در بافر روی هم نوشته شده و از دست خواهد رفت. پیامد این دو پدیده آن بود که در اترنت گیگابیت از کنترل جریان (Flow Control) پشتیبانی شود. (همچنان که در اترنت سریع نیز کنترل جریان وجود دارد ولیکن بروشی متفاوت)

برای عملیات کنترل جریان یکی از طرفین با ارسال یک فریم کنترلی خاص ، از طرف مقابل خود می‌خواهد که برای مدتی ارسال داده را متوقف نماید. فریمهای کنترلی، فریمهای معمولی اترنت هستند که در فیلد Type آنها عدد 0x8808 قرار گرفته است. در این صورت، دو بایت ابتدائی از فیلد داده نوع فرمان را مشخص می‌کند و بایت‌های بعدی به عنوان پارامتر آن فرمان تلقی می‌شوند (در صورت وجود). برای کنترل جریان، فریم PAUSE به کار می‌رود و پارامتر آن مدت زمان توقف را (بر مبنای طول حداقل فریم) مشخص می‌نماید. در اترنت گیگابیت واحد زمان ۵۱۲ نانو ثانیه است و فریم PAUSE می‌تواند ایستگاه را تا ۳۳/۶ میلی‌ثانیه متوقف کند.

پس از آنکه اترنت گیگابیت استاندارد شد کمیتهٔ ۸۰۲ که خسته شده بود می‌خواست پی کار خود برود ولی IEEE از آنها خواست که کار را برروی اترنت ده گیگابیت شروع کنند. پس از جستجوی سخت برای حرفی که بتواند جایگزین z [در 802.3z] شود آنها به این نتیجه رسیدند که از پسوند دو حرفی استفاده کنند!! کار آنها نتیجه داد و استاندارد مربوطه با نام 802.3ae به تایید IEEE رسید. آیا اترنت صد گیگابیت بر ثانیه نیز می‌تواند محقق شود؟

۴-۳-۹ IEEE 802.2: کنترل منطقی لینک

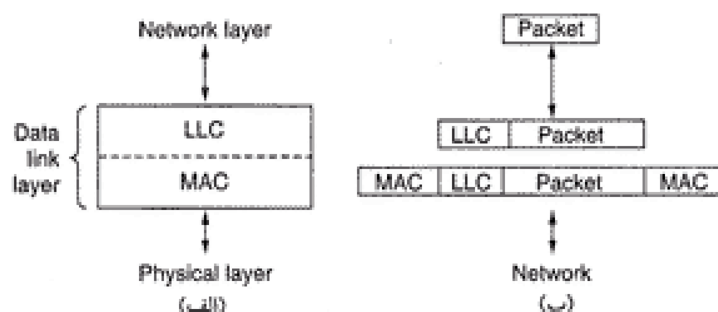
شاید زمان آن فرا رسیده باشد تا اندکی به عقب بازگردیم و برآنچه که در این فصل و ماقبل آن آموختیم، مروری تطبیقی و مقایسه‌ای داشته باشیم. در فصل ۳ آموختیم که چگونه دو ماشین می‌توانند با استفاده از پروتکلهای پیوند داده، بر روی خطی غیرقابل‌اعتماد، به صورت مطمئن مبادلهٔ اطلاعات کنند. این پروتکلها امکان نظارت برخطا (با استفاده از تصدیق دریافت فریم ـ Ack ـ) و کنترل جریان (با استفاده از پنجرهٔ لغزان) را فراهم آورده‌اند.

برعکس، در این فصل در خصوص مبادلهٔ مطمئن داده‌ها سخنی به میان نیاوردیم. تمام گونه‌های اترنت و دیگر پروتکلهای ۸۰۲ تلاش می‌کنند سرویس دیتاگرام عرضه کنند. [یعنی ارسال داده‌ها بدون تصدیق دریافت آنها ـAckـ] در اغلب موارد، این سرویس کافی به نظر می‌رسد. به عنوان مثال برای انتقال بسته‌های IP ، به هیچ تضمینی در رسیدن بسته‌ها نیازی نیست. یک بسته IP را می‌توان بسادگی درون فیلد حمل داده از فریم ۸۰۲ قرار داد و آنرا ارسال کرد. اگر فریم از بین برود مهم نیست.

علیرغم این، سیستمهائی وجود دارند که به پروتکلی با قابلیتهای نظارت برخطا و کنترل جریان نیاز دارند. IEEE پروتکلی تعریف کرده که می‌تواند بر روی پروتکل اترنت یا هر پروتکل سری ۸۰۲ قرار گرفته و اجرا شود. این پروتکل که LLC (Logical Link Control) نامیده می‌شود، قادر است تفاوتهای انواع مختلف شبکه‌های ۸۰۲ را از طریق تعریف یک قالب و واسط (Interface) واحد و مشترک مخفی کند. این پروتکل بسیار شبیه به پروتکل HDLC است که در فصل ۳ آنرا بررسی کردیم. LLC نیمهٔ بالائی لایهٔ پیوند داده‌ها را تشکیل می‌دهد، در حالیکه زیرلایهٔ MAC ، نیمهٔ پائینی این لایه محسوب می‌شود. (شکل ۴-۲۴)

استفادهٔ رایج از LLC بدین ترتیب است: لایهٔ شبکه در ماشین فرستنده، بسته‌ای را بکمک توابع پایهٔ و بنیادی لایهٔ LLC ، بدان تحویل می‌دهد. زیرلایهٔ LLC سرآیند لازم را به بسته می‌افزاید؛ این سرآیند شامل شمارهٔ ترتیب (Seq No) و شمارهٔ تصدیق (Ack No) است. بستهٔ حاصل ، درون فیلد حمل داده از فریم ۸۰۲ قرار گرفته و ارسال می‌شود. در گیرنده نیز عکس این فرآیند انجام می‌شود.

LLC از سه ردهٔ خدمات حمایت می‌کند: (۱) خدمات ارسال نامطمئن دیتاگرام، (۲) خدمات دیتاگرام باتصدیق وصول و (۳) خدمات ارسال اتصال‌گرای مطمئن. سرآیند LLC سه فیلد را در برمی‌گیرد: «نقطه دسترسی در مقصد»، «نقطهٔ دسترسی در مبداء» و «فیلد کنترل». نقطهٔ دسترسی مشخص می‌کند که این فریم از چه پروسه‌ای آمده و به کدام پروسه باید تحویل شود که در حقیقت نقش همان فیلد Type در فریم DIX را ایفاء می‌کند. فیلد کنترل نیز در برگیرندهٔ شمارهٔ ترتیب و شمارهٔ تصدیق است که شباهت فراوانی با ساختار HDLC دارد (شکل ۳-۲۴) ، ولی کاملاً با آن یکسان نیست. از این فیلدها زمانی استفاده می‌شود که در سطح لایهٔ پیوند داده‌ها به یک اتصال مطمئن (Reliable Connection) نیاز باشد که در این حالت شبیه به روشی عمل می‌شود که در فصل سوّم تشریح شد. برای شبکهٔ اینترنت، صرف تلاش در تحویل بستهٔ IP کفایت می‌کند و به هیچ پیغام اعلام وصول فریم در سطح LLC نیازی نیست.



شکل ۴-۲۴. (الف) موقعیت LLC در پشتهٔ پروتکلی (ب) قالبهای پروتکل.

۴-۳-۱۰ نگاهی به گذشتهٔ اترنت

اترنت برای حدود ۲۰ سال در صحنه بوده و تقریباً هیچ رقیب جدّی نداشته است و به نظر می‌رسد در سالهای آتی نیز همچنان یکّه‌تاز باشد. تعداد بسیار کمی سیستم عامل، زبان برنامه نویسی یا معماری CPU وجود داشته که بتواند برای دو دههٔ متوالی برقلهٔ افتخار بایستد و در حال ورود به دههٔ سوّم افتخار خود باشد. روشن است که اترنت حرفهائی برای گفتن داشته که بدین گونه دوام آورده است؛ اینها چه بوده‌اند؟

شاید دلیل اصلی بقای اترنت «سادگی» و «قابلیت اعتماد» آن بوده است. در عمل، معیار «سادگی» به «قابلیت اعتماد»، «ارزانی»، «سهولت نصب و نگهداری» تعبیر می‌شود. وقتی در اترنت انشعابات تزریقی (Vampire Tap) با کانکتورهای BNC عوض شد خرابیها به شدت کاهش یافت. بطور معمول، عموم افراد حاضر نیستند از ابزارهایی که بخوبی کار می‌کنند رو بگردانند و این پافشاری از آنجاست که بر همه روشن شده بسیاری از محصولات بنجل در صنعت کامپیوتر باهیاهو می‌آیند ولی بسیار ضعیف عمل می‌کنند، حتی گاهی محصولاتی که با عنوان «ارتقاء» (Upgrade) معرفی می‌شوند بسیار بدتر و ناسازگارتر از وقتی عمل می‌کنند که بطور کامل عوض شوند! ولیکن نسخه‌های ارتقایافتهٔ اترنت، اعتماد عمومی را جلب کرد.

«سادگی» [در مورد اترنت] به ارزان بودن نیز تعبیر می‌شود. اترنت نازک و سیم‌های زوجی نسبتاً کم بها و ارزان هستند. کارت‌های واسط اترنت نیز بسیار ارزانند. فقط در دورانی که هاب و سوئیچ معرفی شد، به مقداری سرمایه‌گذاری نیاز داشت ولیکن در زمان عرضهٔ آنها، شبکهٔ اترنت بقدر کافی جا افتاده بود.

اترنت از لحاظ نصب و نگهداری ساده است. هیچ نرم‌افزار اضافه‌ای نباید نصب شود (مگر نرم افزارهای بسیار کوچک راه‌انداز آنها) و به هیچ جدول یا تنظیمات پیکربندی خاصی نیاز ندارد که مدیریت آن (یا هرگونه اشتباه در تنظیم آن) کار را دشوار کند. در ضمن اضافه کردن یک ماشین جدید به شبکه به سادگی وصل آن به هاب یا سوئیچ است و کار چندانی ندارد.

امتیاز دیگر اترنت آن بود که بسادگی با TCP/IP که پروتکل غالب دنیاست، کار می‌کند. IP پروتکلی بدون‌اتصال (Connectionless) و دقیقاً متناسب با اترنت است که آن هم بدون اتصال عمل می‌کند. در مقابل، IP سازگاری بسیار کمی با ATM (که اتصال‌گراست) داشت و این عدم سازگاری به موقعیت ATM لطمهٔ بسیار فراوانی زد.

در آخر آنکه اترنت ظرفیت پیشرفت و بهبود فراوانی از خود نشان داد. سرعت آن چند ده برابر شد، هاب و سوئیچ عرضه گردید ولی با تمام این تغییرات نیازی به تغییر در نرم‌افزار نبود. تصور کنید وقتی یک عرضهٔ کنندهٔ محصولات شبکه، تجهیزات مفصلی را ارائه کرده و می‌گوید: «این شبکهٔ جدید و عالی را برای شما تدارک دیده‌ایم. برای استفاده از آن باید زحمت بکشید و سخت‌افزارهای قبلی خود را دور ریخته و تمام نرم‌افزارهای خود را از نو بنویسید!! او در فروش شبکهٔ خود قطعاً به مشکل برخواهد خورد!!! اترنت چنین مشکلی نداشت. شبکه‌هائی مثل FDDI، Fibre Channel و ATM در زمان عرضه بسیار سریعتر از اترنت بودند ولیکن هیچکدام از آنها با اترنت سازگار نبودند، بسیار پیچیده‌تر و مدیریت آنها دشوارتر از اترنت بود. سرانجام وقتی سرعت اترنت بهبود یافت این شبکه‌ها دیگر هیچ مزیتی نداشتند و سریعاً از میان رفتند؛ البته بجز ATM که آن هم در هستهٔ سیستمهای تلفنی به کارگرفته شده بود.

۴-۴ شبکه‌های محلی بی‌سیم

اگر چه اترنت در سطح گسترده‌ای رایج است ولی رقیب جدیدی برای آن در حال ظهور است. شبکه‌های بی‌سیم

در حال رواج هستند و بطور فزاینده‌ای در دفاتر اداری، فرودگاه‌ها و دیگر مکانهای عمومی به کار گرفته می‌شوند. شبکه‌های بی‌سیم به نحوی که در شکل ۱-۳۵ دیدیم به دو روش پیکربندی می‌شوند: «در کنار یک ایستگاه ثابت» و «بدون ایستگاه ثابت». استاندارد 802.11 هر دوی این پیکربندیها را مدّ نظر قرار داده و برای هر دو، تدارک لازم را دیده است.

در بخش ۱-۵-۴ پیش‌زمینه‌ای از 802.11 ارائه نمودیم. حال زمان آن رسیده تا نگاهی دقیقتر به این تکنولوژی بیندازیم. در بخشهای آتی نگاهی به پشته پروتکلی، تکنیکهای ارسال رادیوئی در لایهٔ فیزیکی، پروتکل زیرلایهٔ MAC، ساختار فریم و خدمات ارائه شده در این شبکه خواهیم انداخت. برای کسب آگاهی بیشتر در خصوص 802.11 به مراجع زیر مراجعه نمایید:

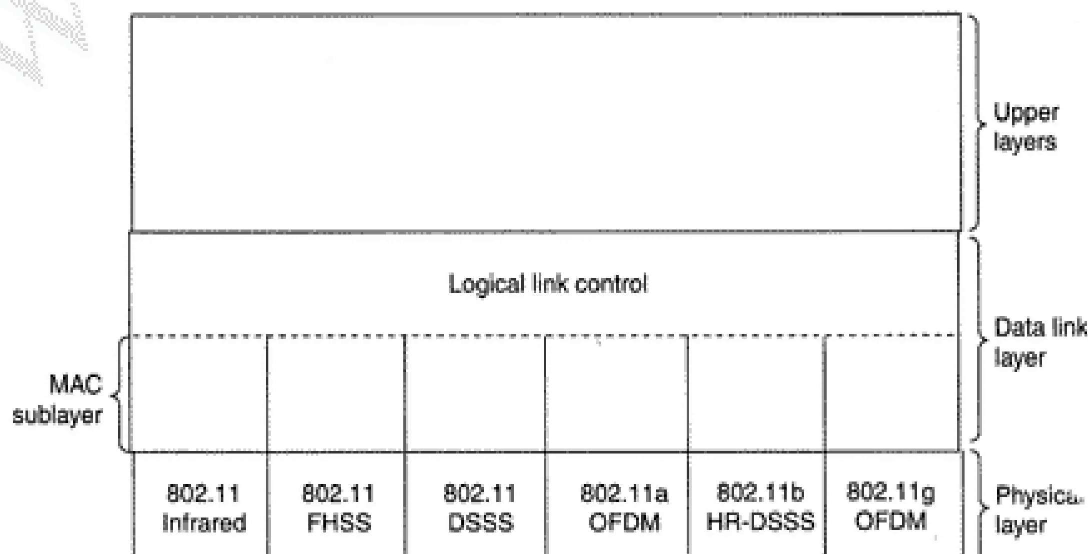
(Crow et al., 1997; Geier, 2002; Heegard et al., 2001; Kapp, 2002; O'Hara & Petric, 1999; and Severance, 1999)

برای شنیدن حقیقت درست و مسلّم این شبکه، باید مستقیماً به استاندارد IEEE 802.11 مراجعه نمایید.

۴-۴-۱ پشتهٔ پروتکلی 802.11

پروتکلهایی که در استانداردهای سری ۸۰۲ و از جمله اترنت به کار گرفته شده مشترکات ساختاری فراوان دارند. شمای کلّی پشتهٔ پروتکلی 802.11، در شکل ۴-۲۵ نمایش داده شده است. لایهٔ فیزیکی به خوبی هم تراز با لایهٔ فیزیکی از مدل OSI است در حالیکه لایهٔ پیوند داده از دو لایهٔ مستقل تشکیل شده است. در 802.11 زیرلایهٔ MAC (زیرلایهٔ کنترل دسترسی به کانال)، چگونگی دسترسی و تخصیص کانال، یعنی ایستگاهی را که باید در ادامه ارسال داشته باشد، مشخص می‌کند. برروی آن، زیرلایهٔ LLC قرار می‌گیرد که وظیفه اصلی آن مخفی کردن تفاوتهای موجود درگونه‌های مختلف ۸۰۲ است،بگونه‌ای که این تفاوتها برای لایهٔ شبکه [لایهٔ سوم در مدل OSI] مخفی و غیر قابل تشخیص باشد. در همین فصل وقتی اترنت را بررسی می‌کردیم، LLC را نیز مطالعه نمودیم و نیازی به تکرار آنها نیست.

در سال ۱۹۹۷ استاندارد 802.11 سه تکنیک مختلف انتقال رادیویی، برای به کارگیری در لایهٔ فیزیکی معرفی کرد. مثلا روش مبتنی بر امواج مادون قرمز، بسیار شبیه به روشی است که در کنترل از راه دور تلویزیونها به کاررفته



شکل ۴-۲۵. بخشی از پشتهٔ پروتکلی 802.11.

است. در روش دیگر از امواج رادیوئی برد کوتاه و از تکنیک‌هائی به نام FHSS و DSSS بهره گرفته شده است. هر دوی این روشها از محدوده‌ای در طیف فرکانس استفاده می‌کنند که نیازی به اخـذ مجوز از دولت نـدارد (بـاند 2.4 GHZ). به عنوان مثال دربازکن‌های کنترل از راه دور نیز از همین باند فرکانسی استفاده می‌کنند لذا کامپیوتر کیفی شما ممکن است در حین استفاده از کانال، خودش را رقیب درب گاراژ شما ببینند!!! تـلفن‌های بـی‌سیم و اجاق‌های مایکروویو نیز از همین باند فرکانسی بهره گرفته‌اند.

تکنیک‌های ارسال رادیوئی با نرخ ۱ تا ۲ مگابیت در ثانیه و با توان بسیار کمی عمل می‌کنند تا تداخل رادیوئی این ابزارها با یکدیگر حداقل باشد. در سال ۱۹۹۹ دو تکنیک جدید معرفی شد تا پهنای باند (نـرخ ارسـال) آن افزایش یابد. این دو تکنیک جدید OFDM و HR-DSSS نامیده شده‌اند و به ترتیب با سرعتهای ۵۴ و ۱۱ مگابیت بر ثانیه عمل می‌کنند. در سال ۲۰۰۱ گونهٔ دوّمی از مدولاسیون OFDM ولی در باند فرکانسی متفاوت نسبت به OFDM اولیه، معرفی شد. در ادامه بطور مختصر آنها را بررسی خواهیم کرد. از دیدگاه فنی این تکنیکها به لایهٔ فیزیکی تعلق دارند و باید در فصل دوّم بررسی می‌شدند ولی از آنجایی که این تـکنیکها بـه LAN و خـصوصاً زیرلایهٔ 802.11 MAC وابسته هستند ، در این بخش بدانها پرداخته‌ایم.

۴-۴-۲ لایهٔ فیزیکی در 802.11

تمام پنج تکنیک انتقال رادیویی، این امکان را فراهم کرده‌اند که یک فریم MAC از ایستگاهی به ایـستگاه دیگـر منتقل شود. تفاوتهای آنها در تکنولوژی به کاررفته و سرعت قابل حصول آنهاست. پرداخـتن بـه جـزئیات ایـن روشها از حوصلهٔ این کتاب خارج است ولیکن چند کلمه صحبت در مورد آنها و بخصوص معرفی کلمات کلیدی آن می‌تواند به علاقمندان کمک کند تا بتوانند برای کسب آگاهی بیشتر، در اینترنت یا مراجع دیگر جستجو کنند.

در گزینهٔ «امواج مادون قرمز» از امواج پخشی (Diffused) با طول موج ۰/۸۵ یا ۰/۹۵ میکرون بهره گرفته شده است. در این روش سرعتهای ۱ و ۲ مگابیت در ثانیه مجاز می‌باشد. در نرخ 1 Mbps از روش کدینگ خاصی به نام Gray Code استفاده شده که در آن گروه‌های ۴ بیتی به یک کلمهٔ کد ۱۶ بیتی تبدیل می‌شوند، به نحوی که در این کلمهٔ ۱۶ بیتی تنها یک بیت ۱ و پانزده بیت ۰ وجود دارد. این کد دارای این ویژگی اساسی است که خطائی کوچک در سنکرونیزاسیون زمان، تنها یک بیت خطا در خروجی ایجاد خواهد کرد. در سرعت 2 Mbps ، دو بیت اخذ و یک کد چهار بیتی تولید می‌شود که در آن تنها یک بیت ۱ وجود دارد. (یعنی هر دو بیت به یکی از چهار حالت 0001 , 0010 , 0100 , 1000 نگاشته می‌شود.) سیگنال‌های مادون قرمز نمی‌توانند در موانعی مثل دیوار نفوذ کنند لذا سلولهائی که در اتاقهای مختلف ایجاد می‌شوند کاملاً از هم جدا و تفکیک‌شده هستند. علیرغم این، بدلیل نرخ ارسال پائین (و این حقیقت که نور خورشید، امواج مادون قرمز را در خود غرق و محو می‌کند) از این گزینه استقبال چندانی نشد.

در FHSS (Frequency Hopping Spread Spectrum) از ۷۹ کانال مستقل استفاده شده که هر یک از این کانالها 1 MHz پهنای باند دارند و از پائینترین فرکانس باند 2.4 GHz ISM شروع می‌شوند. برای مشخص کردن دنبالهٔ فرکانسهائی که باید بدانها پرش شود از یک مولد اعداد شبه‌تصادفی استفاده شـده است. مـادامیکه تـمام ایستگاه‌ها در الگوریتم مولد اعداد از نقطهٔ شروع (Seed) یکسانی استفاده کنند و از لحاظ زمانی با هم سنکرون باشند همگی بطور همزمان به فرکانسهای یکسانی پرش خواهند کرد. مدت زمانی که ایستگاه‌ها در یک فرکانس خاصّ باقی می‌مانند، اصطلاحاً dwell time نامیده می‌شود و پارامتری قابل تنظیم است ولیکن باید کمتر از ۴۰۰ میلی‌ثانیه باشد. استفادهٔ تصادفی از باندهای فرکانسی، روش مناسبی برای تخصیص کانال بروشی غیرمعمول در باند ISM است. این ویژگی کم و بیش به امنیت اطلاعات کمک خواهد کرد چرا کـه اگـر یک اخـلالگر تـرتیب پرشهای فرکانس یا پارامتر Dwell time را نداند، نمی‌تواند اطلاعات کانال را استراق سمع کند. در فواصل دور،

محوشدگی سیگنال بدلیل «چندمسیره شدن سیگنال» (Multipath Fading) اشکال عمده‌ای به حساب می‌آید و خوشبختانه FHSS در مقابله با این مشکل، موفق عمل می‌کند. همچنین این روش نسبت به تداخل رادیوئی، نسبتاً حساس نیست و برای ایجاد لینک بین ساختمانها بسیار مناسب خواهد بود. بزرگترین اشکال این روش پهنای باند کم آنست.(1 Mbps)

سومین روش مدولاسیون یعنی DSSS (Direct Sequence Spread Spectrum) نیز به یکی از نرخ‌های ۱ یا ۲ مگابیت برثانیه محدود شده است. تکنیک به کار رفته در DSSS تا حدودی مشابه به سیستم CDMA است که در بخش ۲.۶.۲ بررسی شد ولیکن از برخی جهات با آن تفاوتهایی دارد. هر بیت در قالب یازده Chips ارسال می‌شود که به دنبالهٔ بارکر (Barker Sequence) مشهور است. در این روش از مدولاسیون تغییر فاز (Phase Shift) با نرخ تغییر 1 Mbaud استفاده شده که وقتی در نرخ 1 Mbps عمل می‌کند در هر تغییر فاز یک بیت ولی وقتی در نرخ 2 Mbps عمل می‌کند در هر تغییر فاز، دو بیت منتقل می‌گردد. برای سالهای متمادی سازمان FCC [سازمان تخصیص فرکانس] فقط اجازه می‌داد که تجهیزات مخابرات بی‌سیم در ایالات متحده، صرفاً از باند ISM و طیف گسترده (Spread Spectrum) استفاده کنند ولی ظهور تکنولوژیهای جدید، به لغو این قانون در سال ۲۰۰۲ انجامید.

اوّلین شبکهٔ محلی بی‌سیم پرسرعت یعنی 802.11a ، با بهره‌گیری از مدولاسیون OFDM[۱] در باند فرکانسی 5-GHz عمل می‌کرد تا به سرعت 54 Mbps دست یابد. اگر بخواهیم با استعارات FDM سخن بگوئیم، در OFDM از ۵۲ زیرکانال فرکانسی استفاده شده که ۴۸ تا از آنها برای داده و ۴ تا برای سنکرونیزاسیون کاربرد دارد و بی‌شباهت به ADSL نیست. از آنجایی که ارسال، بطور همزمان برروی فرکانسهای متفاوتی انجام می‌شود لذا این روش گونه‌ای از روشهای مبتنی بر «طیف گسترده» محسوب می‌شود ولیکن با روشهای CDMA یا FHSS کاملاً متفاوت است. تقسیم سیگنال به تعداد بسیار زیادی باند باریک در مقایسه با استفاده از یک باند عریض و واحد، مزایای بسیار مهمی در بردارد که از جمله می‌توان به ایمنی بیشتر در مقابل تداخل امواج باند باریک و امکان استفاده از باندهای غیر مجاور (noncontiguous band) اشاره کرد. در این روش از سیستم کدینگ پیچیده‌ای مبتنی بر مدولاسیون تغییر فاز برای سرعت زیر 18 Mbps و مدولاسیون QAM برای سرعت‌های بالاتر استفاده شده است. در سرعت 54 Mbps ، ۲۱۶ بیت داده به سمبول‌های ۲۸۸ بیتی کد می‌شود. بخشی از انگیزه‌های خلق OFDM سازگاری آن با سیستم اروپائی HiperLAN/2 بوده است. (Doufexi et al., 2002) این روش دارای کارآئی بسیار بالائی در استفاده از طیف فرکانسی (برحسب bits/HZ) بوده و ایمنی خوبی در مقابل پدیدهٔ «محوشدگی ناشی از مسیرهای چندگانه» دارد.

نهایتاً به HR-DSSS (High Rate Direct Spread Spectrum) می‌رسیم که روشی دیگر مبتنی بر تکنیک طیف گسترده است و با به کارگیری 11 million chips/sec در باند 2.4 GHz ، به نرخ ارسال یازده مگابیت در ثانیه رسیده است. این روش 802.11b نامیده شده ولیکن دنباله روی 802.11a نبوده است. در این روش از نرخهای ۱، ۲، ۵/۵ و ۱۱ مگابیت در ثانیه حمایت می‌شود. دو نرخ ارسال پائین [یعنی ۱ و ۲ مگابیت بر ثانیه] به ترتیب از سیگنالی با نرخ تغییر 1 Mbaud/sec بهره می‌گیرد (یعنی در هر تغییر فاز، ۱ یا ۲ بیت کد می‌شود). در HR-DSSS به منظور سازگاری با DSSS ، از روش مدولاسیون تغییر فاز بهره گرفته شده است. برای نرخ ارسال سریعتر (۵/۵ و ۱۱ مگابیت بر ثانیه) از سیگنالی با نرخ تغییر 1.375 Mbaud/sec استفاده شده و در هر تغییر به ترتیب ۴ یا ۸ بیت کد می‌شود؛ کدها از نوع Walsh/Hadamard هستند. نرخ ارسال به صورت پویا و در خلال عملیات شبکه تعیین

---

۱. Orthogonal Frequency Division Multiplexing

می‌شود تا سرعت بهینه بر اساس شرایط فعلی حاکم بر شبکه (شامل نویز محیط و بار) تنظیم گردد. در عمل، سرعت شبکه 802.11b تقریباً همیشه 11 Mbps است. اگر چه سرعت 802.11b از سرعت 802.11a کمتر می‌باشد ولیکن بُرد این شبکه حدوداً هفت برابر بیشتر است که این ویژگی در بسیاری از محیطها اهمیت بسزائی دارد.

نسخهٔ بهبود یافتهٔ 802.11b یعنی 802.11g ، در نوامبر سال ۲۰۰۱ (پس از کشمکش فراوان بر سر آنکه از کدام تکنولوژی استفاده شود) به تائید IEEE رسید. این استاندارد از مدولاسیون OFDM (به کار رفته در 802.11a) بهره می‌گیرد ولیکن مثل 802.11b در باند 2.4 GHz عمل می‌کند. از نظر تئوری این سیستم می‌تواند در سرعت 54 Mbps عمل کند، ولیکن هنوز روشن نیست که آیا این سرعت در عمل نیز محقق خواهد شد یا خیر. بدین ترتیب کمیتهٔ 802.11 سه شبکهٔ محلی پرسرعت بی‌سیم معرفی کرده است. 802.11a ، 802.11b و 802.11g (به سه شبکهٔ کندتر فعلاً کاری نداریم) شاید این سوال درست به ذهن‌شما خطور کند که آیا تعریف سه استاندارد متفاوت کار درستی است؟ شاید عدد ۳، عدد طلائی شانس باشد!

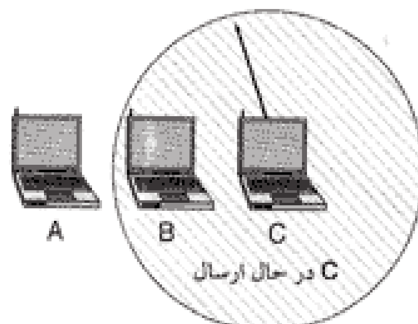
۳-٤-٤ پروتکل زیرلایهٔ MAC در 802.11

حال اجازه بدهید از فضای مهندسی برق به سرزمین مهندسی کامپیوتر برگردیم. پروتکل زیرلایهٔ MAC در 802.11 کاملاً با اترنت تفاوت دارد زیرا شرایط حاکم بر محیطهای بی‌سیم در مقایسه با سیستمهای سیم‌دار، دارای پیچیدگیهای ذاتی است. در اترنت یک ایستگاه منتظر می‌ماند تا کانال آزاد شود؛ سپس ارسال خود را شروع می‌کند. اگر در خلال ارسال ۶۴ بایت اوّل فریم، هیچ نویز شدیدی برنگشت، می‌توان اعتقاد داشت که فریم بـه درستی تحویل مقصد شده است. در شبکهٔ بی‌سیم چنین وضعیتی حاکم نیست.

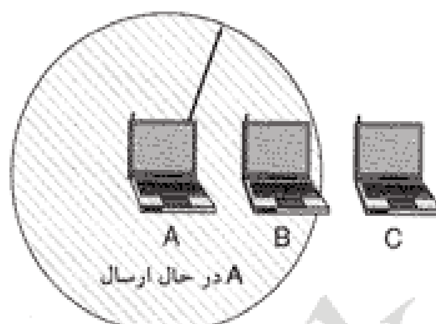
برای شروع باز هم یادآور می‌شویم که مشکل ایستگاه مخفی (که قبلاً تشریح و مجدداً در شکل ۴-۲۶ به تصویر کشیده شده است) ایجاد اشکال خواهد کرد. از آنجایی که تمام ایستگاه‌ها در برد رادیوئی یکدیگر نیستند فلذا ارسال سیگنال در بخشی از یک سلول، ممکن است در ناحیهٔ دیگری از همان سلول قابل دریافت و شنود نباشد. در این مثال ایستگاه C در حال ارسال به ایستگاه B است ولی اگر A کانال را بررسی و شنود کند هیچ چیزی نمی‌شنود و به غلط نتیجه می‌گیرد که باید ارسال برای B را آغاز کند. [شکل ۴-۲۶-الف]

همچنین عکس این مشکل نیز وجود دارد. در شکل ۴-۲۶-ب ایستگاه B می‌خواهد فریمی را برای C بفرستد و به همین دلیل به کانال گوش می‌دهد. وقتی سیگنال در حال انتقال را شنود می‌کند به غلط نتیجه می‌گیرد که نباید برای ایستگاه C چیزی بفرستد ولی علیرغم آنکه A در حال ارسال برای D است (D در شکل نشان داده نشده) ایستگاه B می‌تواند برای C ارسال داشته باشد. مضاف براین، ارتباط بی‌سیم اغلب ماهیت «دوطرفهٔ غیرهمزمان» (Half Duplex) دارد بدین معنا که ایستگاه‌ها نمی‌توانند در حین ارسال و بطور همزمان، کانال را برای آگاهی از وضعیت تصادم برروی همان باند فرکانسی شنود کنند. [این کار در کانالهای سیمی به راحتی امکان‌پذیر است] در نتیجه 802.11 ، برخلاف اترنت از CSMA/CD استفاده نمی‌کند.

برای رفع این مشکلات، استاندارد 802.11 از دو روش عملکرد پشتیبانی می‌کند: در اولین روش که DCF نامیده می‌شود (Distributed Coordination Function) هیچ‌گونه کنترلی مرکزی وجود ندارد (و از این دیدگاه مشابه با اترنت است). در روش دیگر که PCF (Point Coordination Function) نامیده می‌شود، برای کنترل و نظارت بر کلیهٔ فعالیتهای درون هر سلول ، از یک ایستگاه ثابت استفاده شده است. در پیاده‌سازی استاندارد 802.11، باید از DCF پشتیبانی شود در حالیکه حمایت از PCF اختیاری است. به ترتیب این دو روش را تشریح خواهیم کرد.



(الف)



(ب)

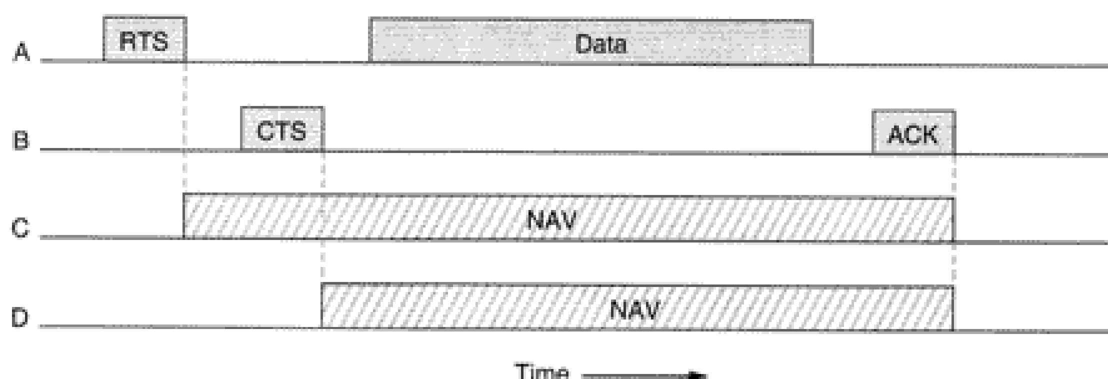
شکل ۴-۲۶. (الف) مشکل ایستگاه مخفی (ب) مشکل ایستگاه آشکار.

وقتی از حالت DCF استفاده می‌شود، 802.11 از پروتکلی به نام CSMA/CA[۱] بهره می‌گیرد. در این پروتکل هم کانال فیزیکی و هم کانال مجازی شنود می‌شوند. CSMA/CA از دو عملکرد متفاوت پشتیبانی می‌کند: در روش اوّل وقتی یک ایستگاه می‌خواهد فریمی را ارسال کند، ابتدا به شنود می‌پردازد و اگر کانال آزاد بود ارسال خود را آغاز می‌کند. در حین ارسال فریم، **کانال شنود نمی‌شود** و کلّ فریم منتقل می‌گردد، در حالیکه ممکن است این فریم، بدلیل تداخل رادیوئی در گیرنده از بین برود. برعکس، اگر کانال اشغال باشد فرستنده ارسال خود را تا زمان آزاد شدن کانال به تعویق انداخته و سپس شروع می‌کند. در صورت بروز تصادم، ایستگاه‌های تصادم‌کننده به اندازهٔ یک زمان تصادفی (که بر اساس الگوریتم عقب‌گرد نمایی تعیین می‌شود) منتظر مانده و از نو تلاش می‌کنند. [تا اینجا همه چیز شبیه به پروتکل CSMA/CD است.]

روش دیگر به کاررفته در CSMA/CA مبتنی بر MACAW است که از روش «شنود کانال مجازی» بهره می‌گیرد. در مثال شکل ۴-۲۷، ایستگاه A می‌خواهد فریمی را برای B بفرستد. C، ایستگاهی در برد ایستگاه A است (شاید C هم در برد ایستگاه B باشد ولی مهم نیست). D، ایستگاهی در برد B است ولی در برد A قرار ندارد. پروتکل زمانی آغاز می‌شود که A تصمیم به ارسال داده برای B می‌گیرد. او کارش را با ارسال یک فریم کوتاه RTS برای B آغاز و تقاضای مجوز ارسال فریم می‌نماید. هرگاه B این تقاضا را دریافت کند (احتمالاً تصمیم به صدور مجوز می‌گیرد؛ در این حالت فریم CTS را برمی‌گرداند. پس از دریافت فریم CTS، ایستگاه A ارسال فریم خود را شروع کرده و یک زمان‌سنج خاصّ به نام ACK-Timer را روشن می‌کند. پس از دریافت فریم داده، ایستگاه B با ارسال فریم ACK به مبادله داده خاتمه می‌دهد. اگر زمان‌سنج، قبل از آنکه فریم ACK باز گردد، منقضی شود [یعنی فریم ACK در زمان معقولی برنگردد] کلّ این فرآیند باید از نو اجرا شود.

حال بیائید از دیدگاه ایستگاه‌های C و D به این فرآیند مبادلهٔ فریم، نگاه کنیم. C ایستگاهی است که در برد A است و طبعاً RTS را دریافت می‌کند. اگر اینگونه باشد متوجه خواهد شد که شخص دیگری بزودی ارسال خود را آغاز خواهد کرد، فلذا برای احتیاط کامل، تا تکمیل عملیات مبادلهٔ داده، از ارسال هر چیزی اجتناب می‌کند. C می‌تواند از طریق اطلاعاتی که از فریم RTS بدست می‌آید، کلّ زمان ارسال را تخمین بزند (با احتساب زمان برگشت فریم ACK)، لذا برای خودش فرض می‌کند که در خلال زمانی که در شکل ۴-۲۷ با عنوان NAV

۱.CSMA with Collision Avoidance



شکل ۴ـ۲۷. کاربرد کانال مجازی در روش CSMA/CA.

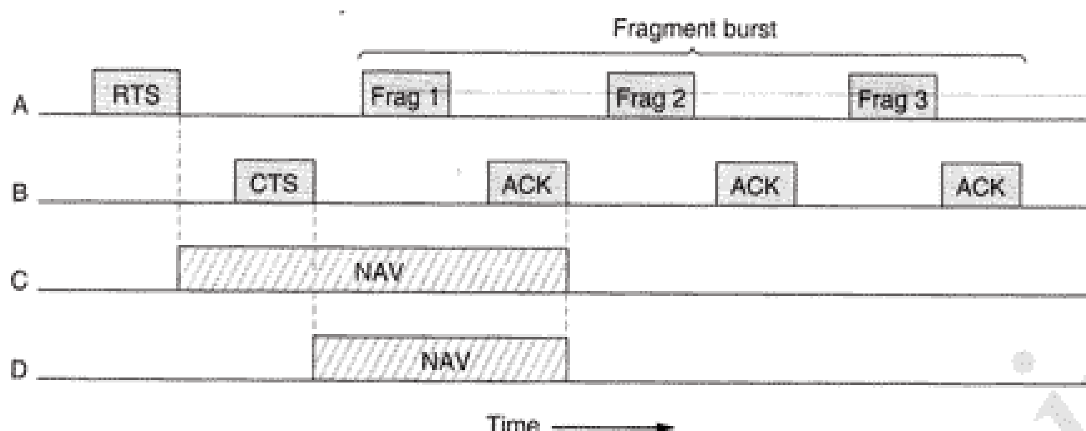
(Network Allocation Vector) مشخص شده، کانال مجازی مشغول است. ایستگاه D فریم RTS را نمی‌شنود ولیکن CTS را خواهد شنید لذا او هم برای خودش به اندازه زمان NAV کانال را مشغول فرض می‌کند. دقت کنید که سیگنال NAV به هیچ وجه ارسال نمی‌شود بلکه فقط یک یادداشت داخلی است که ایستگاه را در مدت زمان معینی ساکت نگه می‌دارد.

برخلاف شبکه‌های سیمی، شبکه‌های بی‌سیم غیرقابل‌اعتماد و نویزی هستند چراکه دستگاه‌های مختلف مثل اجاقهای مایکروویو (که آنها نیز در باند ISM کار می‌کنند) نویز قوی و مخرب تولید می‌نمایند. در نتیجه ، احتمال انتقال موفق فریم با افزایش طول فریم کاهش خواهد یافت. هرگاه احتمال خرابی یک بیت p باشد احتمال آنکه یک فریم n بیتی، کامل و سالم دریافت شود $(1-p)^n$ خواهد بود. برای مثال اگر $p=10^{-4}$ باشد احتمال سالم رسیدن یک فریم کامل اترنت (۱۲۱۴۴ بیتی) کمتر از ۳۰ درصد است. برای $p=10^{-5}$ ، از هر ۹ فریم یکی خراب خواهد شد. برای $p=10^{-6}$ حدود یک درصد از کل فریمها آسیب خواهد دید که تقریباً معادل یک دوجین فریم در ثانیه خواهد بود. بطور خلاصه اگر یک فریم بزرگ باشد شانس کمتری در سالم رسیدن آن به مقصد وجود دارد و احتمالاً باید مجدداً ارسال شود.

برای کاهش مشکل کانالهای نویزی، 802.11 اجازه داده که هر فریم به قطعات کوچکتری تقسیم شده و هر کدام کد کشف خطای خود را داشته باشند. قطعات بطور مجزا شماره‌گذاری شده و دریافت آن به روش **«توقف و انتظار»** (Stop & Wait) تائید می‌شود. (به عبارت دیگر فرستنده قطعه شمارهٔ k+1 را نخواهد فرستاد مگر آنکه پیام ACK قطعهٔ kام ـمبنی بر دریافت صحیح آن‌ـ دریافت شود.) پس از آنکه به کمک RTS و CTS ، کانال در اختیار ایستگاهی قرار گرفت، طبق شکل ۴–۲۸ ، آن ایستگاه می‌تواند متوالیاً چندین قطعهٔ مستقل ، ارسال کند. دنبالهٔ قطعات متوالی، اصطلاحاً Fragment Burst نامیده می‌شود.

عملیات قطعه‌قطعه‌سازی فریمها، کارائی مفید شبکه را افزایش خواهد داد چرا که به جای ارسال مجدد کل فریم فقط قطعات کوچکی که در اثر خطای کانال خراب شده‌اند از نو ارسال خواهند شد. طول هر قطعه به صورت قطعی و ثابت در استاندارد تعیین نشده است بلکه جزو پارامترهای قابل تنظیم هر سلول محسوب و توسط ایستگاه ثابت (Base Station) تنظیم می‌شود. مکانیزم NAV ایستگاه‌ها را آنقدر ساکت نگاه می‌دارد تا زمانیکه دریافت کلّ فریم تایید شود، ولیکن برای آنکه ایستگاه‌ها آنقدر تامل کنند تا دنبالهٔ کل قطعات فریم (Fragment Burst) بدون تداخل ارسال شود ، مکانیزم دیگری که در زیر تشریح شده، بکار می‌رود.

تمام توضیحات فوق در حالت DCF از استاندارد 802.11 قابل اعمال و صائب است. یادآوری می‌کنیم که در حالت DCF هیچ کنترل و نظارت مرکزی وجود ندارد و تمام ایستگاه‌ها مشابه با آنچه که در اترنت اتفاق می‌افتد برای بدست آوردن کانال رادیویی رقابت می‌کنند. در حالت دیگر یعنی PCF ، یک ایستگاه ثابت یکی‌یکی به



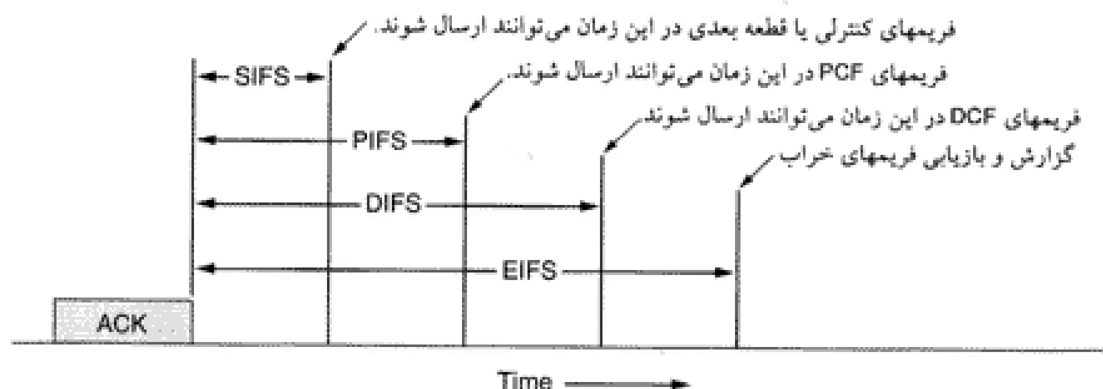
شکل ۴ـ۲۸. ارسال انفجاری چند قطعه.

ایستگاه‌ها سرکشی کرده و از آنها سوال می‌کند که آیا فریمی جهت ارسال دارند یا خیر. از آنجایی که در حالت PCF بر تمام تقاضاهای ارسال فریم به صورت مرکزی نظارت می‌شود لذا هیچگونه تصادم اتفاق نخواهد افتاد. استاندارد 802.11 ، مکانیزم سرکشی به ایستگاه‌ها را تعیین کرده است ولیکن دفعات سرکشی ، ترتیب سرکشی یا حتی تصمیم‌گیری در خصوص آنکه آیا همهٔ ایستگاه‌ها باید به یک اندازه از شبکه سرویس بگیرند، در استاندارد تعریف و تعیین نشده است.

مکانیزم اصلی در سرکشی به ایستگاه بدین نحو است که یک ایستگاه فریم خاصی به نام Beacon Frame (فریم فانوس دریائی) را بطور متناوب در فضای پیرامون خود منتشر می‌کند. (ده تا صد بار در هر ثانیه) «فریم فانوس دریائی» شامل پارامترهای مختلف سیستم مثل: «ترتیب پرش فرکانسی» (Hopping Sequence) و پارامتر dwell time (برای مدولاسیون FHSS) و پارامتر سنکرون‌سازی سیگنال ساعت و نظایر آن، می‌باشد. همچنین توسط این فریم از ایستگاه‌های جدید دعوت می‌شود تا به منظور سرکشی شدن ثبت‌نام کنند. پس از آنکه ایستگاهی برای دریافت خدمات سرکشی با نرخ معین، ثبت‌نام کرد، این تضمین را دارد که بخش معینی از پهنای باند شبکه به او اختصاص داده خواهد شد و بدین ترتیب آن ایستگاه می‌تواند **«کیفیت خدمات»** (Quality of Service) خود را تضمین کند. [برای مطالعه در خصوص کیفیت خدمات به بخش ۱-۳-۳ و فصل ۵ مراجعه کنید.]

طول عمر باطری یکی از مسائل مهم در ابزارهای همراه و بی‌سیم بوده و هست؛ به همین دلیل در استاندارد 802.11 به مسئله مدیریت توان مصرفی، توجه ویژه‌ای شده است. خصوصاً در حالت PCF ایستگاه ثابت می‌تواند ایستگاه همراه را به «حالت استراحت» (Sleep State) ببرد؛ تا زمانیکه بطور مشخص ایستگاه ثابت یا کاربری دیگر آن را از این حالت بیرون آورده و بتواند فعالیت عادّی خود را از سربگیرد. مجبور کردن یک ایستگاه به استراحت، بدین معناست که ایستگاه ثابت مسئولیت دارد تمام فریمهائی را که برای ایستگاه غیرفعال (در حال استراحت) ارسال می‌شود، دریافت و بافر کند. بعداً این فریمها به صورت یکجا تحویل خواهد شد.

در درون یک سلول می‌توان بطور همزمان هم حالت PCF و هم حالت DCF را به کار گرفت. در نگاه اوّل ممکن است به نظر برسد که نظارت مرکزی و نظارت توزیع‌شده بطور همزمان میسّر نباشد ولی در استاندارد 802.11 برای رسیدن به چنین هدفی راهکاری مناسب اندیشیده شده است. برای این کار، بازه‌های زمانی بین فریمها بدقت تعریف می‌شود. پس از آنکه یک فریم ارسال شد و قبل از آنکه ایستگاهی بتواند فریم بعدی را ارسال نماید به مدّتی **«زمان مرده»** نیاز است. در این زمان مرده، چهار بازهٔ زمانی مجزا با اهداف خاص، تعریف



شکل ۴ـ۲۹. فاصلهٔ زمانی بین فریمها در 802.11.

شده است. این چهار بازهٔ زمانی در شکل ۴-۲۹ نشان داده شده است.

کوتاه‌ترین بازهٔ زمانی، بازهٔ SIFS (Short InterFrame Spacing) است. این بازهٔ زمانی به ایستگاه‌ها فرصت می‌دهد تا به ارسال فریمهای کنترلی خاص بپردازند. در این زمان ایستگاه‌ها اجازه می‌یابند عملیاتی مثل ارسال CTS (در پاسخ به RTS)، ارسال فریم ACK در پاسخ به یک فریم کامل یا یک قطعه از فریم، ارسال یک قطعه از دنبالهٔ قطعات (بدون ارسال RTS مجدّد) یا نظائر این را انجام بدهند.

همیشه فقط یک ایستگاه است که پس از زمان SIFS، به منظور پاسخ‌دهی [و ارسال فریم کنترلی مناسب] حقّ ارسال دارد. اگر ایستگاه مربوط نتواند از این فرصت استفاده کند و زمان PIFS (PCF InterFrame Spacing) منقضی شود، ایستگاه ثابت می‌تواند «فریم فانوس دریائی» (Beacon) یا «فریم سرکشی» ارسال نماید. این مکانیزم اجازه می‌دهد که ایستگاه در حال انتقال فریم یا دنبالهٔ قطعات، بدون آنکه ایستگاه دیگری در این میان مداخله کند ارسال فریم خود را به پایان برساند در حالیکه ایستگاه ثابت نیز این فرصت را خواهد داشت که وقتی ایستگاه قبلی کارش را به اتمام رساند کانال را بدون رقابت با کاربران متمایل به ارسال تصرف نماید.

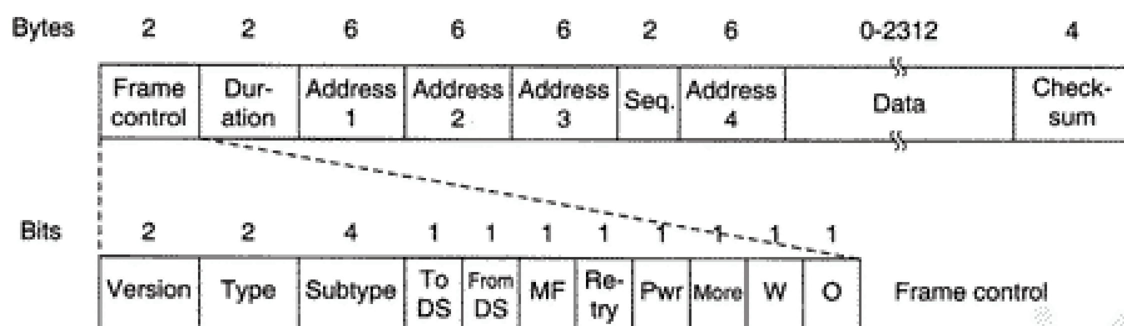
هرگاه ایستگاه ثابت چیزی برای ارسال نداشته باشد و زمان DIFS (DCF InterFrame Spacing) منقضی شود هر ایستگاه می‌تواند بخت خود را در تصرف کانال و ارسال فریم بیازماید. در این لحظه، برای در اختیار گرفتن کانال، روش معمولی رقابت و در صورت تصادم الگوریتم عقب‌گرد نمایی اعمال می‌شود.

آخرین بازهٔ زمانی یعنی EIFS (Extended InterFrame Spacing) مورد استفادهٔ ایستگاهی قرار می‌گیرد که یک فریم خراب یا فریمی ناشناس دریافت کند تا بتواند این مسئله را گزارش بدهد. دلیل اصلی آنکه به این بازهٔ زمانی کمترین اولویت داده شده [آخرین بازهٔ زمانی است] آن بوده که چون گیرندهٔ نمی‌داند چه اتفاقی در جریان است لذا باید مدت زمان قابل توجهی صبر کند تا از هرگونه مداخله در گفتگوی دو ایستگاه اجتناب نماید.

٤.٤.٤ ساختار فریم 802.11

استاندارد 802.11 سه ردهٔ مختلف فریم برای ارسال برروی کانال تعریف کرده است: **«فریم داده»**، **«فریم کنترلی»** و **«فریم مدیریتی»**. هر یک از این فریمها سرآیند (Header) خاصّ خود را دارند که در هر سرآیند، فیلدهائی جهت استفاده در زیرلایهٔ MAC تعریف شده است. مضاف بر این سرآیندهائی جهت به کارگیری در لایهٔ فیزیکی تعریف گردیده که اغلب تکنیک‌های مدولاسیون [و پارامترهای مخابراتی] را مشخص می‌کنند، لذا در اینجا بدانها نخواهیم پرداخت.

قالب فریم داده، در شکل ۴-۳۰ نشان داده شده است. در ابتدا «فیلد کنترل» ظاهر شده که این فیلد خودش دارای یازده فیلد فرعی است. اولین زیرفیلد، شمارهٔ نسخهٔ پروتکل (Protocol Version) را مشخص می‌کند؛



شکل ۴ـ۳۰. فریمهای داده در 802.11.

بدین ترتیب در آن واحد و در یک سلول مشابه، به کارگیری دو پروتکل متفاوت ممکن خواهد بـود. در ادامه، زیرفیلد دو بیتی Type آمده که نوع فریم را [اعم از فریم داده، کنترلی و مدیریتی] مشخص می‌نماید؛ پس از آن زیرفیلد Subtype قرار گرفته که مشخصات دقیقتر فریم (مثل RTS و CTS) را تعریف مـی‌کند. سپس بیتهای ToDS و FromDS آمده که مشخص می‌کنند که آیا فریم از یک «سیستم توزیع درون‌سلولی» (مثلاً شبکه اترنت) بیرون آمده یا بدانجا رهسپار است. بیت MF نشان می‌دهد که هنوز قطعاتی از فریم در پیش رو هستند [و هنوز قطعاتی از فریم باقیست.]بیت Retry نشانگر آنست که فریم جاری قبلاً یکبار ارسال شده است. ایستگاه ثابت بکمک بیت Power Management (که در شکل با نماد Pwr نشان داده شده است)، ایستگاه متحرک را به حالت استراحت (Sleep) برده یا آنرا از حالت استراحت بیرون می‌آورد. بیت More نشان می‌دهد که فرستنده باز هم فریمهائی برای ارسال به گیرنده آماده دارد. بیت W مشخص می‌کند که بدنهٔ فریم با استفاده از الگوریتم WEP (Wire Equivalent Privacy) (فصل نهم) رمزنگاری شده است. نهایتاً بیت O به گیرنده تفهیم می‌کند دنباله‌ای از فریمها که این بیت در آنها ۱ است باید الزاماً به ترتیب (و پشت سرهم) پردازش شوند.

فیلد دوم از فریم داده یعنی فیلد Duration ، مشخص‌کنندهٔ آنست که ارسال فریم جاری و دریافت ACK آن، جمعاً چه زمانی کانال را به حالت اشغال در خواهد آورد. این فیلد که در فریمهای کنترلی نیز وجود دارد مشخص می‌نماید که ایستگاه‌های دیگر به چه نحوی باید مکانیزم NAV خود را مدیریت کنند. سرآیند فریم حاوی چهار آدرس است که همگی منطبق با قالب استانداردی هستند که توسط IEEE 802 تعریف شده است. بدیهی است که به دو فیلد آدرس مبداء و مقصد نیاز بوده ولیکن دوتای دیگر چه کاربردی دارند؟ به خاطر داشته باشید که یک فریم ممکن است از طریق ایستگاه ثابت به یک سلول وارد یا از سلول خارج شود. دو آدرس اضافی، برای تعیین ایستگاه ثابت مبداء و مقصد بکار می‌رود. (وقتی که ترافیک داده‌ها بین چند سلول در حال جریان است).

فیلد Sequence اجازه می‌دهد تا قطعات یک فریم شماره‌گذاری شوند. از شانزده بیت موجود در این فیلد ۱۲ بیت، هویت فریم را و ۴ بیت شمارهٔ قطعه را مشخص می‌کند. در فیلد Data داده‌هـای خام قـرار مـی‌گیرد که می‌تواند حداکثر ۲۳۱۲ بایت باشد. در آخر نیز فیلد Checksum قرار گرفته که به مـنظور کـشف خـطا کـاربرد دارد.

«فریم مدیریت» قالبی مشابه با «فریم داده» دارد با این تفاوت که این فریم یکی از آدرسهای ایستگاه ثابت را ندارد، زیرا فریمهای مدیریتی فقط محدود به یک سلول خاص هستند [و بین سلول‌های متفاوت مبادله نخواهند شد.] فریم کنترلی بازهم کوتاهتر هستند و فقط یک یا حداکثر دو فیلد آدرس دارند. این فریمها، فاقد فیلد داده و فیلد Sequence هستند. اطلاعات اساسی اینگونه فریمها، درون فیلد Subtype نهفته است که عموماً نوع RTS ، CTS یا ACK را مشخص می‌کند.

۵-٤-٤ خدمات

استاندارد 802.11 بیان داشته که هر شبکه محلی بی‌سیم باید ۹ نوع خدمات عرضه نماید. این خدمات به دو رده تقسیم بندی شده‌اند. پنج نوع خدمات «**توزیعی**» و چهار نوع خدمات «**ایستگاهی**». خدمات توزیعی در خصوص مدیریت بر عضویت ایستگاههای درون سلول و تعامل با ایستگاه‌های خارج از سلول است. در مقابل، خدمات ایستگاهی صرفاً در خصوص فعالیتهای درون یک سلول واحد ارائه می‌شود.

پنج نوع خدمات توزیعی که توسط ایستگاه‌های ثابت عرضه می‌شوند ، در خصوص قابلیت تحرّک ایستگاه‌های همراه ، ورود و خروج آنها به سلولها و اتصال یا انفصال از ایستگاه ثابت کاربرد دارند. این خدمات عبارتند از:

۱. Association (**پیوستن به شبکه**): ایستگاه‌های متحرک از این سرویس بهره می‌گیرند تا خود را به ایستگاه ثابت متصل نمایند. بطور معمول زمانی که یک ایستگاه متحرک وارد محدودهٔ رادیوئی یک ایستگاه ثابت می‌شود با استفاده از این سرویس، هویت و قابلیتهای خود را معرفی می‌کند. قابلیتها عبارتند از نرخ ارسال داده‌ها (نرخ‌های متعددی که از آن حمایت می‌شود)، نیاز به خدمات متعدّد PCF (مثل سرکشی) و نیازمندیهای آن در خصوص مدیریت توان مصرفی. ایستگاه ثابت می‌تواند حضور ایستگاه متحرک را بپذیرد یا ردّ کند. اگر ایستگاه متحرک پذیرفته شد بایستی هویت خود را اثبات کند. [روشهای احراز هویت را در فصل نهم مطالعه کنید.]

۲. Disassociation (**ترک شبکه**): ممکن است ایستگاه متحرک یا ایستگاه ثابت، در هر زمان اراده کنند از یکدیگر جدا شوند و ارتباط خود را قطع نمایند. وقتی ایستگاهی بخواهد شبکه را ترک کند یا خاموش شود، از این سرویس بهره می‌گیرد اما ایستگاه ثابت نیز قبل از قطع موقّت ارتباط ممکن است از آن استفاده کند.

۳. Reassociation (**پیوستن مجدد**): یک ایستگاه متحرک می‌تواند با استفاده از این سرویس، ایستگاه ثابت خود را تغییر بدهد. این قابلیت زمانی کاربرد دارد که ایستگاه‌های متحرک بخواهند از یک سلول به سمت سلول دیگر حرکت نمایند. اگر از این امکان به موقع و صحیح استفاده شود در اثر این جابجائی، هیچ داده‌ای از دست نخواهد رفت.

۴. Distribution (**توزیع**): این سرویس مسیر ارسال فریمهای ارسالی به سوی ایستگاه ثابت را تعیین می‌نماید. اگر مقصد فریمها در همان محل ایستگاه ثابت واقع شده باشد فریم می‌تواند مستقیماً از طریق هوا ارسال شود. در غیر این صورت فریمها باید از طریق شبکهٔ سیمی (ارتباط سیمی بین ایستگاه‌های ثابت) به مقصد هدایت شود.

۵. Integration (**یکپارچگی**): اگر نیاز باشد فریمی به شبکه‌ای غیر از 802.11 (با ساختار آدرس و ساختار فریم متفاوت) ارسال شود، این سرویس می‌تواند وظیفه ترجمه و تبدیل قالب فریم (متناسب با شبکه مقصد) را برعهده بگیرد.

چهار سرویس باقیمانده در درون یک سلول بکار می‌آیند. (بعبارت دیگر در خصوص عملیات درون یک سلول واحد کاربرد دارند.) این سرویسها پس از پیوستن ایستگاه‌ها به یک ایستگاه ثابت [به عنوان عضوی از شبکه] مورد استفاده قرار گرفته و به عبارت زیر هستند:

۱. Authentication (**احراز هویت**): از آنجایی که در مخابرات بی‌سیم ایستگاه‌های غیرمجاز و بیگانه نیز می‌توانند ارسال و دریافت داشته باشند فلذا هر ایستگاه باید قبل از دریافت مجوز ارسال، هویت خود را اثبات کند. پس از آنکه یک ایستگاه متحرک به عنوان عضو ، به یک ایستگاه ثابت پیوست (یا بعبارت دیگر درون

سلول پذیرفته شد)، ایستگاه ثابت یک فریم خاص به نام «فریم چالش» (Challenge) برای او می‌فرستد تا ببیند آیا آن ایستگاه کلید سرّی (کلمه عبور) خود را می‌داند یا نه؟ ایستگاه، آگاهی خود از کلید سرّی را با رمز کردن فریم و بازگرداندن آن، اثبات می‌نماید. اگر نتیجه درست باشد، ایستگاه متحرک در درون سلول ثبت‌نام و عضو می‌شود. در استاندارد اولیه لازم نبود که ایستگاه ثابت نیز عضویت خود را احراز کند در حالیکه اصلاح این اشکال بزرگ در دست اجرا است.

۲. Deauthentication **(لغو حضور در شبکه)**: وقتی یک ایستگاه که قبلاً احراز هویت (و عضو) شده بخواهد شبکه را ترک کند باید لغو حضور خود در شبکه را به اطلاع ایستگاه ثابت برساند. پس از لغو حضور و ترک شبکه، ایستگاه دیگر نمی‌تواند از شبکه بهره بگیرد.

۳. Privacy **(محرمانه نگاه داشتن اطلاعات)**: برای محرمانه ماندن اطلاعاتی که از طریق شبکهٔ بی‌سیم مبادله می‌شود، بایستی رمز شوند. این کار از طریق رمزنگاری و رمز گشائی میّسر است. الگوریتم رمزنگاری مورد استفاده روش RC4 می‌باشد که توسط رونالد ری‌وست در دانشگاه MIT ابداع شده است.

۴. Delivery **(تحویل)**: انتقال داده‌ها کل آن هدفی است که در پی آن هستیم لذا در 802.11 روشی برای ارسال و دریافت داده‌ها ارائه شده است. از آنجایی که 802.11 مبتنی بر مدل شبکهٔ اترنت است و مبادله اطلاعات در اترنت تضمین صددرصد ندارد، 802.11 نیز مبادله مطمئن داده‌ها را تضمین نمی‌کند. لایه‌های بالاتر باید در خصوص کشف و تصحیح خطا کاری انجام بدهند.

در استاندارد 802.11 ، هر سلول پارامترهائی دارد که می‌توان آنها را بررسی و در صورت نیاز تنظیم کرد. این پارامترها در خصوص عملیات رمزنگاری ، نرخ ارسال ، بازه‌های زمانی [زمان‌های انقضای مهلت یا Timeouts]، تناوب ارسال فریمهای Beacon ونظایر اینها، تعریف شده‌اند.

شبکه‌های محلی بی‌سیم 802.11 ، در حال ورود به عرصه ادارات ، فرودگاه‌ها ، هتلها، رستورانها و محوطه‌های دانشگاهی هستند و انتظار می‌رود رشد سریع و چشمگیری داشته باشند. برای آنکه در خصوص استفادهٔ گسترده از 802.11 ، اطلاعات و تجاربی بدست بیاورید به مرجع (Hills, 2001) مراجعه کنید.

۴ـ۵ بی‌سیم با باند گسترده

بیش از اندازه به درون (شبکه‌های LAN) پرداخته‌ایم. حال اجازه بدهید پا را فراتر گذاشته و ببینیم آیا در خارج از دنیای LAN هم شبکه‌های جالبی وجود دارند. با رفع موانع قانونی از مقررات حاکم بر عرضهٔ خدمات تلفن، در بسیاری از کشورها شرکتهای مخابراتی رقیب اجازه یافتند تا به ارائه خدمات صوت و خدمات اینترنت پرسرعت بپردازند. امروزه نیاز عمومی در این خصوص بسیار بالا و پررونق است. برای چنین شرکتهایی مسئله اساسی آنست که کشیدن کابلهای فیبرنوری، کابل کوآکسیال یا سیمهای زوجی (از نوع CAT 5) تا درب در میلیونها خانه و محل کار، بسیار گران و نامعقول به نظر می‌رسد. پس این رقبای تجاری برای عرضه خدمات ارزان، چه باید می‌کردند؟

پاسخ به این نیاز **«شبکهٔ بی‌سیم‌باند گسترده»** است. برافراشتن یک آنتن بزرگ برروی قله یا یک تپه و نصب آنتنهائی برروی پشت بام مشتریان که به سمت آنتن اصلی جهت‌گیری شده، بسیار ارزان‌تر از حفر زمین و کابل‌کشی است. بهمین دلیل شرکتهای مخابرات راه دور بیشتر گرایش دارند که برای عرضه خدمات اینترنت ، ارسال صدا و ارائه نمایشهای ویدیوئی راه دور، از سیستمهای چندگیگابیتی بی‌سیم بهره بگیرند. همانگونه که در شکل ۲ـ۳۰ ملاحظه نمودید، LMDS به همین منظور ابداع شد ولیکن تاکنون هر یک از شرکت‌های عرضه‌کنندهٔ

LMDS ، به سلیقهٔ خود سیستمی را طراحی و نصب کرده است. فقدان یک استاندارد واحد، بدین معناست که سخت‌افزار و نرم‌افزار آنها نمی‌تواند بطور انبوه تولید شود و طبعاً قیمتها بالا و استقبال عمومی از آنها کم خواهد بود.

نهایتاً بسیاری از دست اندرکاران صنعت بدین نتیجه رسیدند که داشتن یک استاندارد برای بی‌سیم باند گسترده [با نرخ ارسال بسیار بالا] یکی از مولفه‌های اصلی و حلقهٔ مفقوده در کار آنهاست، لذا از IEEE خواستند تا برای تدوین چنین استانداردی ، کمیته‌ای با حضور دست‌اندرکاران شرکتهای مهم و مراکز دانشگاهی، تشکیل بدهد. اولین عدد اختصاصی داده نشده در ردیف شماره‌های 802.x ، ۱۶ بود و به همین دلیل استاندارد ، با عنوان IEEE 802.16 معرفی شد. کار کمیته در ژولای ۱۹۹۹ شروع و استاندارد نهائی در آوریل ۲۰۰۲ به تائید رسید. نام رسمی استاندارد «**واسط هوائی برای سیستمهای بی‌سیم غیرمتحرک با پهنای باند وسیع**»[۱] انتخاب شده ولیکن برخی از افراد ترجیح می‌دهند آنرا «شبکه بین‌شهری بی‌سیم» یا «حلقهٔ بی‌سیم» بنامند. ما تمام این واژه‌ها را معادل یکدیگر فرض خواهیم کرد.

همانند بسیاری از استانداردهای دیگر سری ۸۰۲، استاندارد 802.16 نیز شدیداً تحت تاثیر مدل OSI بوده و این تاثیر در (زیر) لایه‌ها، اصطلاحات، سرویسهای پایه و موارد دیگر بخوبی محسوس است. متاسفانه این استاندارد نیز شبیه به OSI پیچیده شده است. در بخش بعدی بطور مختصر نکات و ویژگیهای برجستهٔ 802.16 را برخواهیم شمرد ولی این توضیحات به هیچ وجه کامل نبوده و جزئیات آن ناگفته خواهد ماند.

۴ـ۵ـ۱ مقایسهٔ 802.11 با 802.16

در همان ابتدا ممکن است بدین نکته بیندیشید که چرا استاندارد جدیدی ابداع شد؟ چرا از 802.11 استفاده نشد؟ دلائل متعدد و محکمی برای عدم استفاده از 802.11 وجود دارد. در اصل 802.11 و 802.16 نیازهای متفاوتی را برآورده می‌کنند. قبل از پرداختن به فناوری 802.16 شاید چند کلمه‌ای بحث در خصوص دلائل نیاز به استاندارد جدید، خالی از لطف نباشد.

محیطی که 802.11 و 802.16 در آن، عمل می‌کنند از چند منظر شبیه به هم هستند: در اصل هر دوی این استانداردها بدان جهت طراحی شده‌اند که ارتباط بی‌سیم با پهنای باند بسیار بالا را میسّر نمایند. ولیکن این دو شبکه از جهات بسیار مهمی با هم متفاوتند. اصلی‌ترین تفاوت آنست که 802.16 برای ارائه خدمات به ساختمانها طراحی شده است و طبعاً ساختمانها حرکت نمی‌کنند!!! ساختمانها تغییر سلول نمی‌دهند! بخش اعظم عملیات 802.11 با مسائل ناشی از متحرک بودن ایستگاه‌ها سروکار دارد و چنین مسائلی در 802.16 ، محلّی از اعراب ندارد. گذشته از آن، در ساختمانها ممکن است بیش از یک کامپیوتر وجود داشته باشد و پیچیدگیهای چنین محیطی در جائی که ایستگاه نهائی یک کامپیوتر کیفی واحد است، بروز نخواهد کرد. از آنجایی که مالکین ساختمان معمولاً آمادگی پرداخت پول بیشتری در مقایسه با صاحب یک کامپیوتر کیفی دارند لذا می‌توان خدمات ارتباط رادیوئی بهتری در اختیارشان گذاشت. این تفاوت بدین معناست که در 802.16 می‌توان ارتباط دوطرفهٔ همزمان (Full Duplex) داشت در حالیکه برای پائین نگاه داشتن هزینهٔ ارتباط رادیوئی در 802.11 از آن اجتناب شده است. [ارتباط 802.11 دو طرفهٔ غیر همزمان ـ Half Duplex ـ و با برد بسیار کم می‌باشد.]

از آنجایی که 802.16 در محدودهٔ بخشی از یک شهر به اجرا در می‌آید، فواصل [بین نقاط] می‌تواند تا چندین کیلومتر باشد و این موضوع بدان معناست که توان مورد نیاز ایستگاه ثابت می‌تواند بسته به موقعیت و فاصله‌ها متغیر باشد. این تفاوتها و تغییرها برروی نسبت سیگنال به نویز (SNR) تاثیر گذاشته و در نتیجه، باید به اجبار از

۱. Air Interface for Fixed Broadband Wireless Access Systems

چندین روش مدولاسیون استفاده شود. همچنین مخابرهٔ آزادانه در سطح یک شهر مبیّن آنست که تمهیدات امنیتی و حفظ حریم افراد، نیازی بنیادی و اجتناب‌ناپذیر است.

به علاوه، در مقایسه با سلول‌های معمول 802.11 ، در اینجا سلولها می‌توانند کاربران بسیار زیادتری را در بر بگیرند و این کاربران نیز توقع پهنای باند بیشتری نسبت به کاربران 802.11 دارند. گذشته از آن به ندرت اتفاق می‌افتد که شرکتی از پنجاه کارمند خود دعوت کند تا برای مشاهدهٔ اشباع شدن [و از کار افتادن 802.11] دور هم جمع شده و پنجاه فیلم مجزا تماشا کنند!! [در حالیکه در یک ساختمان مسکونی احتمال دارد پنجاه نفر بدیدن فیلم از طریق شبکه مشغول باشند.] به همین دلیل به پهنای وسیعتری از طیف فرکانسی موجود در باند ISM نیاز است و طبعاً 802.16 مجبور است در محدودهٔ فرکانسی ۱۰ تا ۶۶ گیگاهرتزی عمل کند؛ یعنی تنها محدودهٔ استفاده نشده از طیف فرکانس که هنوز موجود است.

ولیکن این امواج با طول موج میلی‌متری نسبت به امواج با طول موج بیشتر در باند ISM ، ویژگی‌های فیزیکی کاملاً متفاوتی دارند و در نتیجه به لایهٔ فیزیکی کاملاً متفاوتی نیاز است. یکی از ویژگی‌های امواج میلی‌متری آنست که توسط آب شدیداً جذب می‌شوند. (بالاخص باران، در برخی از مواقع برف، تگرگ و متاسفانه مه غلیظ) در نتیجه مدیریت خطا بسیار با اهمیت‌تر از محیط‌های داخلی [مثل محیط LAN] است. امواج میلی‌متری می‌توانند به خط مستقیم متمرکز و منتشر شوند (در حالیکه در 802.11 انتشار امواج در همهٔ جهات است) فلذا گزینه‌ها و تمهیدات پیش‌بینی شده در ارتباط با انتشار چندمسیره (Multipath) در 802.16 محلّی از اعراب ندارد.

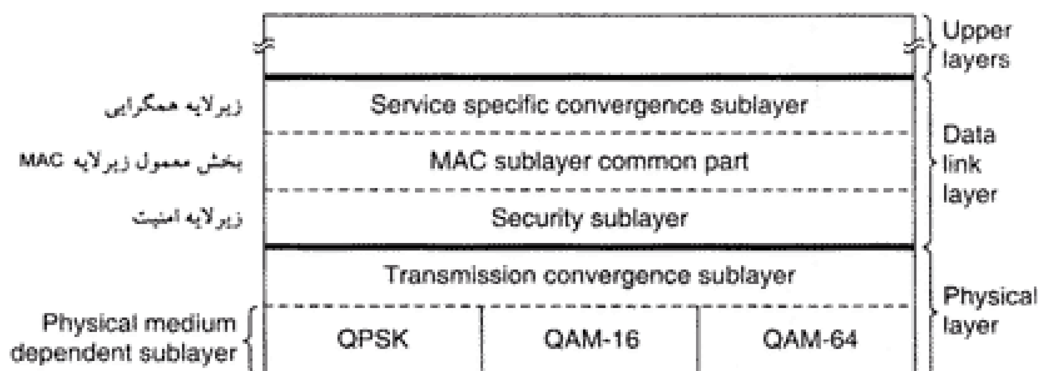
مورد دیگر مربوط به «**کیفیت خدمات**» (QoS) است. اگر چه 802.11 برای ترافیک بی‌درنگ پشتیبانیهائی به عمل آورده (در حالت PCF) ولیکن حقیقتاً برای کاربردهای تلفنی و عملیات چندرسانه‌ای سنگین و دائم طراحی نشده است. در مقابل، از 802.16 انتظار می‌رود که کاملاً از چنین کاربردهائی پشتیبانی نماید زیرا برای استفاده در محیط‌های مسکونی و اداری مدّ نظر بوده است.

کوتاه سخن آنکه 802.11 طراحی شده تا یک اترنت متحرک باشد در حالیکه 802.16 طراحی شده تا شبیه به تلویزیون کابلی، ثابت ولی بی‌سیم عمل نماید. این تفاوتها آنقدر بنیادیست که استانداردهای حاصل اختلاف فراوانی دارند و در هر یک سعی در بهینه‌سازی و حل و فصل نیازهای متفاوتی شده است.

مقایسه‌ای بسیار کوتاه با سیستم تلفن سلولی [تلفن همراه] نیز خالی از فایده نیست. وقتی در مورد تلفن‌های همراه صحبت می‌کنیم روی سخن با ایستگاه‌هائی همراه با محوریت ارسال صوت، توان مصرفی پائین و پهنای باند باریک است که از امواج مایکروویو با طول موج متوسط بهره گرفته‌اند. هیچ‌کس یک فیلم دو ساعته و با کیفیت بالا را برروی گوشی موبایل GSM خود تماشا نمی‌کند (البته فعلاً)!! حتی UMTS نیز امیدی به تغییر چنین وضعیتی ندارد. [امید به افزایش چشمگیر پهنای باند در GSM] در عبارتی کوتاه در دنیای شبکه‌های بین شهری بی‌سیم (Wireless MAN) تقاضا برای پهنای باند بسیار بیشتر از دنیای تلفن همراه است و طبعاً به سیستمهائی کاملاً متفاوت نیاز می‌باشد. اینکه آیا 802.16 می‌تواند در آینده برای ابزارهای متحرک و همراه نیز به کارگرفته شود سوال جالبی است: این شبکه برای چنین محیطی بهینه نشده ولیکن شاید بشود! فعلاً تمرکز آن برروی شبکهٔ دادهٔ بی‌سیم و غیرمتحرک است.

۴-۵-۲ پشتهٔ پروتکلی 802.16

شکل ۴-۳۱ پشتهٔ پروتکلی 802.16 را نشان می‌دهد. ساختار کلّی این پشته، شبیه به شبکه‌های دیگر 802 است ولیکن تعداد بیشتری زیرلایه دارد. پائینترین زیرلایه با انتقال فیزیکی بیتها بر روی کانال سروکار دارد و در آن از رادیوی باند باریک و روش‌های معمول مدولاسیون، بهره گرفته شده است. برروی «لایهٔ فیزیکی انتقال» ، زیرلایهٔ «**همگرائی**» (Convergence Sublayer) قرار گرفته تا تکنولوژیهای متفاوت به کار رفته در زیر را از دید لایهٔ پیوند



شکل ۴ـ۳۱. پشتهٔ پروتکلی 802.16.

داده‌ها مخفی نگاه دارد. در حقیقت، 802.11 نیز چیزی شبیه به همین زیرلایه را دارد ولی کمیتهٔ مربوطه ، آنرا با استفاده از اسامی رایج در مدل OSI تعریف نکرده است.

اگرچه در شکل ۴-۳۱ نشان نداده‌ایم ولی در آینده دو پروتکل جدید به لایهٔ فیزیکی اضافه می‌شود: (۱) استاندارد 802.16a که از روش مدولاسیون OFDM در باند ۲ تا ۱۱ گیگاهرتز پشتیبانی می‌نماید. (۲) استاندارد 802.16b که در باند ۵ گیگاهرتز ISM عمل می‌کند. در هر دوی اینها تلاش شده تا 802.16 به 802.11 نزدیکتر شود.

لایهٔ پیوند داده متشکل از سه زیرلایه است: زیرلایهٔ پائینی با امنیت و محرمانه نگاه داشتن اطلاعات سروکار دارد که برای شبکه‌های عمومی در محیطهای باز بسیار حیاتی‌تر از شبکه‌های خصوصی بسته مثل اترنت است. این زیرلایه عملیات رمزنگاری، رمزگشائی و مدیریت کلیدها را برعهده دارد.

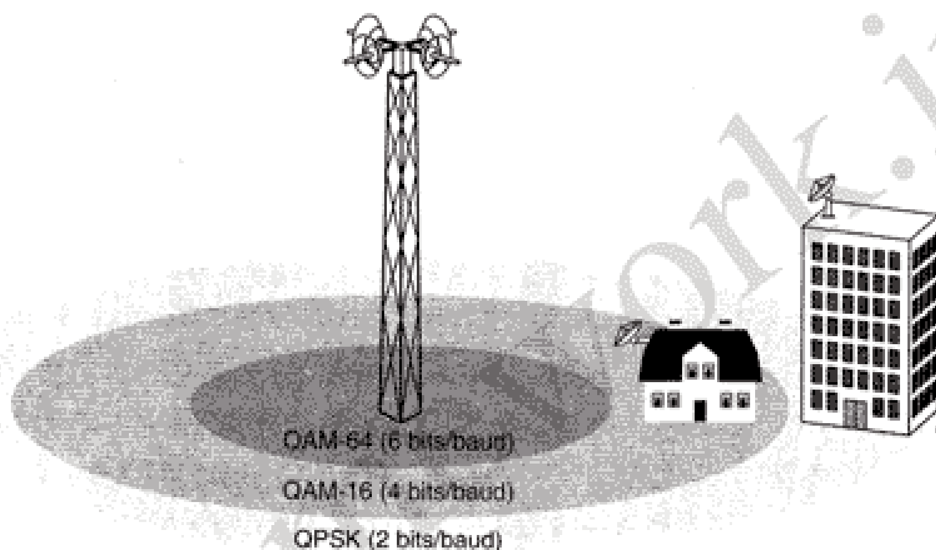
سپس بخش مشترک از زیرلایهٔ MAC قرار می‌گیرد. این لایه همان نقطه‌ای است که پروتکلهای اساسی مثل پروتکلهای مدیریت کانال در بر می‌گیرد. در این مدل مبنا آنست که ایستگاه ثابت ، سیستم را کنترل می‌کند. این ایستگاه می‌تواند «جریان اطلاعات از ایستگاه ثابت به مشترکین» (که اصطلاحاً downstream نام دارد) را به صورت کاملاً متفاوت و متمایز زمان‌بندی نماید و نیز نقش بسیار مهمی در مدیریت «جریان اطلاعات از مشترکین به ایستگاه ثابت» (Upstream) ایفا می‌کند. یکی از ویژگیهای نامتعارف در زیرلایهٔ MAC آنست که برخلاف شبکه‌های دیگر 802 ، این استاندارد کلاً اتصال‌گرا (Connection Oriented) است، تا بتواند «کیفیت خدمات» (QoS) را برای ارتباطات تلفنی و سایر کاربردهای چندرسانه‌ای تضمین نماید.

زیرلایهٔ **«همگرائی خاص سرویس‌دهی»** (Service Specific Convergence Sublayer) به جای «زیرلایهٔ لینک منطقی» (یعنی زیرلایهٔ معمول LLC) قرار گرفته است. عملکرد این زیرلایه ایجاد واسطی متناسب با لایهٔ شبکه است. پیچیدگی این زیرلایه از آنجا نشأت می‌گیرد که 802.16 به نحوی طراحی شده تا بتواند با پروتکلهای دیتاگرام (مثل PPP ، IP و اترنت) و همچنین شبکهٔ اتصال‌گرای ATM قابل جمع باشد و با آنها کار کند. مشکل آنجاست که پروتکلهای «مبتنی بر بسته» بدون‌اتصال (Connectionless) هستند در حالیکه ATM اتصال‌گراست. این بدین معناست که هر اتصال ATM بایستی به یک اتصال در 802.16 نگاشته شود که اصولاً کار ساده و سرراستی است. ولی سوال این است که یک بستهٔ IP ورودی باید برروی کدامین اتصال 802.16 نگاشته شود؟ این مشکل در همین زیرلایه حلّ خواهد شد.

۴ـ۵ـ۳ لایهٔ فیزیکی در 802.16

، بالا اشاره شد که در بی‌سیم با پهنای باند وسیع، ب... حس گسترده‌تری از طیف فرکانسی نیاز است و تنها محل

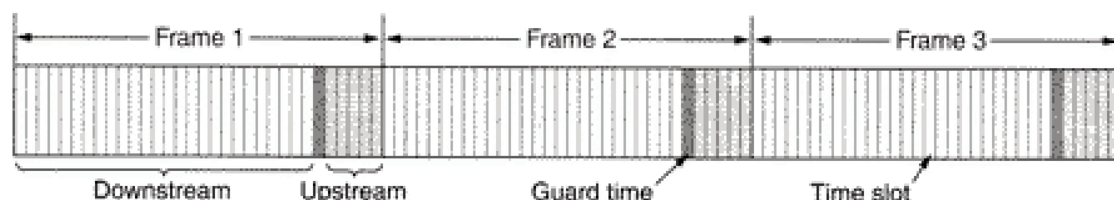
دسترسی به چنین وسعتی محدودهٔ ۱۰ تا ۶۶ گیگاهرتزی است. این امواج با طول‌موج میلی‌متری ویژگیهای جالبی دارند که امواج مایکروویو با طول موج بلندتر ندارند: آنها برخلاف صوت و مشابه با نور به خط مستقیم سیر می‌کنند. در نتیجه، ایستگاه ثابت بایستی چندین آنتن داشته باشد و هر یک از آنها بسوی قطاع خاصی از مناطق پیرامون خود نشانه رفته باشد. (شکل ۴-۳۲) هر قطاع ، کاربران خاصّ خود را دارد و مستقل از قطاعهای همجوار خود است؛ چنین ساختاری برای رادیوی سلولی صادق نیست چرا که آنتنهای آنها «همه‌جهته» (Omnidirectional) هستند.



شکل ۴ـ۳۲. محیط انتقال در 802.16.

از آنجایی که در باند امواج میلی‌متری توان سیگنال براساس فاصله از ایستگاه ثابت، شدیداً کاهش می‌یابد لذ سبت سیگنال به نویز (SNR) نیز برحسب فاصله افت خواهد داشت. به همین دلیل 802.16 بسته به فاصلهٔ یک مشترک از ایستگاه ثابت، سه روش مدولاسیون متفاوت را به کارگرفته است. برای مشترکین نزدیک، از روش QAM-64 با مشخصهٔ 6 bits/baud استفاده شده است. برای مشترکین با فاصلهٔ متوسط از QPSK با مشخصهٔ 2 bits/baud استفاده می‌شود. برای مشترکین دوردست ، روش QPSK با مشخصه 2 bits/boud به کار می‌رود. برای مثال به ازای پهنای معمول ۲۵ مگاهرتزی از طیف فرکانس، پهنای باند روش QAM-64 نرخ ۱۵۰ مگابیت بر ثانیه، روش QAM-16 نرخ ۱۰۰ مگابیت بر ثانیه و QPSK نرخ ۵۰ مگابیت بر ثانیه را در اختیار قرار می‌دهد. به عبارت دیگر هر چه مشترکین از ایستگاه ثابت دورتر باشند، نرخ ارسال پائینتر خواهد بود (شبیه به آنچه که در خصوص ADSL در شکل ۲ـ۲۷ دیدید) در شکل ۲-۲۵ نمودار فضائی این سه مدولاسیون ، نمایش داده شده است.

با توجه به آنکه هدف اصلی ایجاد سیستمی با باند وسیع بوده و با در نظر داشتن محدودیتهای فیزیکی، طراحان 802.16 کوشیدند تا از طیف موجود به نحو بهینه استفاده کنند. آنها به روش به کاررفته در GSM و DAMPS علاقه‌ای نداشتند. هر دوی این سیستمها از فرکانسهای متفاوت ولی در باند مشابهی برای ارسال جریان ترافیک رو به بالا و رو به پائین (Upstream/Downstream) بهره می‌گیرند. برای صوت شاید ترافیک در اکثر بخشها متقارن باشد [بعبارت دیگر برای صوت میزان ارسال و دریافت حدوداً به یک اندازه است] ولیکن برای دسترسی به اینترنت غالباً حجم ترافیک دریافتی بیشتر از ترافیک ارسالی است. در نتیجه 802.16 روشی منعطفتر برای تخصیص پهنای باند ارائه کرده و از دو روش FDD (Frequency Division Duplexing) و TDD (Time Division Duplexing) بهره گرفته است. روش دوّم یعنی TDD ، در شکل ۴-۳۳ نشان داده شده



شکل ۴-۳۳. فریمها و برشهای زمانی در روش TDD (Time Division Duplexing) .

است. در اینجا ایستگاه ثابت بطور متناوب فریمهائی را منتشر می‌کند. هر فریم شامل تعدادی برش زمانی مستقل (Time Slot) است. برشهای ابتدائی هر فریم، برای ارسال ترافیک روبه‌پائین [یعنی از ایستگاه ثابت به کاربر] در نظر گرفته شده است. پس از آن «**زمان مراقبت**» (Guard Time) فرا می‌رسد که به ایستگاه‌ها مهلت می‌دهد تا جهت ارسال و دریافت خود را تغییر بدهند. در آخر برشهائی را برای ارسال ترافیک رو به بالا [از کاربر به ایستگاه ثابت] خواهیم داشت. تعداد برشهای زمانی را که به ارسال در هر یک از جهات، اختصاص می‌یابد، می‌توان به صورت پویا تغییر داد تا پهنای باند در هر جهت، با حجم ترافیک تطبیق داشته باشد.

ترافیک روبه‌پائین توسط ایستگاه ثابت در درون برشهای زمانی نگاشته می‌شود. ایستگاه ثابت بطور کامل بر این جهت از جریان، کنترل دارد. ترافیک روبه‌بالا [که توسط کاربران تولید می‌شود] با پیچیدگی بیشتری مواجه بوده و میزان آن به کیفیت خدمات (QoS) مورد نیاز بستگی دارد. در ادامه وقتی به تشریح زیرلایهٔ MAC پرداختیم به روش تخصیص برشهای زمانی هم خواهیم رسید.

ویژگی جالب دیگر در لایهٔ فیزیکی، توانائی آن در ارسال متوالی و پشت سرهم فریمهای MAC ، در قالب یک انتقال فیزیکی واحد است. این ویژگی سربار ناشی از بیتهای آغازین (Preamble) و سرآیند لازم در لایهٔ فیزیکی را کاهش داده و در نتیجه بازده مفید طیف را افزایش خواهد داد.

نکتهٔ قابل توجه دیگر، استفاده از **کدهای همینگ** (Hamming) به منظور تصحیح مستقیم خطا در لایهٔ فیزیکی است.[۱] تقریباً در تمام شبکه‌ها، به کدهای کشف خطا بسنده می‌شود و هرگاه فریم دریافتی دارای خطا باشد ارسال مجدّد صورت می‌گیرد، ولی از آنجاییکه در محیط‌های باز و در نرخ ارسال بالا، احتمال بروز خطا در حین انتقال، خیلی بیشتر است لذا گذشته از عملیات کشف خطا (که در لایه‌های بالاتر انجام می‌شود) در لایهٔ فیزیکی نیز عملیات تصحیح خطا صورت می‌گیرد. تاثیر نهائی تصحیح خطا آنست که کانال بهتر از آنی که هست به نظر می‌رسد. (بدلیل مشابه اگر چه CD-ROMها قابل‌اعتماد به نظر می‌رسند ولیکن این اعتماد، حاصل از تخصیص بیش از نیمی از بیتها به تصحیح خطا در لایهٔ فیزیکی است.) [به عبارت دیگر در هر CD ، بیش از نیمی از بیتهای سطح دیسک فقط به منظور عملیات کشف و تصحیح خطا ذخیره شده‌اند!]

٤-٥-٤ پروتکل زیرلایهٔ MAC در 802.16

همانگونه که در شکل ۴-۳۱ دیدیم، در 802.16 ، لایهٔ پیوند داده‌ها به سه زیرلایه تقسیم می‌شود. از آنجایی که ما تا فصل هشتم به مطالعه رمزنگاری نخواهیم پرداخت لذا تشریح عملکرد «زیرلایهٔ امنیت» در اینجا سخت است. به همین مقدار بسنده می‌کنیم که برای سرّی نگاه داشتن داده‌های ارسالی از رمزنگاری در سطح لایهٔ فیزیکی و به صورت بی‌درنگ، بهره گرفته شده است. صرفاً بخش دادهٔ هر فریم رمزنگاری می‌شود و سرآیند آن (Header) رمز نخواهد شد. این خصوصیّت بدین معناست که یک جاسوس قادر است بفهمد چه کسی با چه کسی محاوره می‌کند ولی قادر به فهم آنچه با یکدیگر می‌گویند نیست.

---

۱. Forward Error Correction

اگر با رمزنگاری آشنائی قبلی داشته باشید در حدّ یک پارگراف، زیرلایهٔ امنیت را بررسی می‌کنیم ولیکن در صورت عدم آشنائی با رمزنگاری، پارگراف بعدی چیزی به دانش شما نخواهد افزود. (می‌توانید این پارگراف را پس از اتمام فصل ۸ مجدداً مطالعه نمائید.)

زمانی که یک مشترک به ایستگاه ثابت متصل می‌شود، دوبه‌دو یکدیگر را با استفاده از روش «رمزنگاری RSA» و مبتنی بر «گواهینامه‌های X.509» احراز هویت می‌کنند. بخش دادهٔ فریمهای آنها با استفاده از یک سیستم رمزنگاری متقارن رمز می‌شود که این سیستم یا مبتنی بر «DES با زنجیره‌سازی بلوکها» است و یا از روش «DES سه‌گانه با دو کلید» استفاده می‌شود. احتمالاً به زودی روش AES (Rijndael) نیز به آن اضافه می‌شود. عملیات بررسی صحّت داده‌ها نیز با استفاده از SHA-1 انجام می‌گیرد. تا اینجا چندان بد نبود، نظر شما چیست؟!!

حال اجازه بدهید نگاهی به بخش مشترک از زیرلایهٔ MAC بیندازیم. فریمهای MAC ، تعداد مشخصی از برشهای زمانی لایهٔ فیزیکی را اشغال می‌کنند. هر فریم از چندین فریم کوچکتر (Subframe) تشکیل شده که دو تای ابتدائی آن به ترتیب برای نگاشت و حمل ترافیک روبه‌بالا (Upstream) و روبه‌پائین (Downstream) در نظر گرفته شده است. ساختار این نگاشت (یعنی درج داده‌ها درون برشهای زمانی) بگونه‌ای است که مشخص می‌کند چه چیزی در یک برش زمانی درج شده و کدامیک از برشهای زمانی آزاد هستند. نگاشت ترافیک روبه‌پائین نیز شامل پارامترهای مختلف سیستمی است تا شروع فعالیت یک ایستگاه جدید را به اطلاع ایستگاه ثابت برساند.

عملکرد کانال روبه‌پائین نسبتاً ساده و سر راست است. ایستگاه ثابت بسادگی تصمیم می‌گیرد که چه چیزی را در کدام فریم کوچک (Subframe) قرار بدهد. عملکرد کانال رو به بالا پیچیده‌تر است زیرا در اینجا مشترکین رقیب و ناهماهنگ، برای در اختیار گرفتن آن با یکدیگر رقابت می‌کنند. روش تخصیص کانال بـه مـواردی در خصوص «کیفیت خدمات» (QoS) بستگی دارد. در 802.16 چهار رده از خدمات، به ترتیب ذیل تعریف شده‌اند:

۱. **خدمات با نرخ ارسال ثابت** (Constant Bit Rate Service)
۲. **خدمات بی‌درنگ با نرخ ارسال متغیر** (Real-time Variable Bit Rate)
۳. **خدمات غیر بی‌درنگ با نرخ ارسال متغیر** (NonReal-time Variable bit Rate)
۴. **خدمات مبتنی بر «حداکثر تلاش»** (Best Effort Service)

تمام خدمات عرضه شده در 802.16 اتصال‌گرا هستند و به هر اتصال[1] فقط یکی از رده‌های خدمات فوق تعلّق می‌گیرد و نوع خدمات نیز در زمان برقراری اتصال تعیین می‌شود. این طراحی بسیار متفاوت از 802.11 یا اترنت است که در آنها هیچ اتصالی در زیرلایهٔ MAC ایجاد نمی‌شود.

«خدمات با نرخ ارسال ثابت» برای انتقال سیگنال صوتی غیرفشرده همانند یک کانال T1 مدّ نظر بوده است. به این خدمات از آن جهت نیاز است که بتوان حجم معینی از داده‌ها را در زمان مشخصی ارسال کرد. برای رسیدن به این هدف، به هر اتصال از این نوع، تعداد ثابت و معینی برش زمانی (در هر فریم[2]) اختصاص داده مـی‌شود. هرگاه پهنای باند لازم اختصاص داده شود این برشهای زمانی بطور خودکار و بدون نیاز به درخواست بعدی در اختیار خواهد بود.

«خدمات بی‌درنگ با نرخ ارسال متغیر» برای کاربردهای چندرسانه‌ای فشرده‌شده و هرگونه عملیات بی‌درنگ که در آنها مقدار پهنای باند مورد نیاز، متغیر است سودمند خواهد بود. این کار بدین نحو انجام می‌شود که ایستگاه ثابت در فواصل زمانی مشخص به مشترک خود سرکشی کرده و از او در خصوص میزان پهنای باند مورد نیازش

۱. اتصال را یک ارتباط منطقی و هماهنگ شده بین دو نقطه در شبکه در نظر بگیرید. -م

۲. معنای واژهٔ فریم در اینجا با معنای متعارف آن تفاوت دارد. معنای فریم 802.16 در شکل ۴-۳۳ مشخص است. -م

سوال می‌کند.

«خدمات غیربی‌درنگ با نرخ ارسال متغیر» برای حجم ترافیک سنگینی که بی‌درنگ نیست (همانند انتقال فایل‌های طولانی) در نظر گرفته شده است. برای ارائهٔ چنین خدماتی، اغلب ایستگاه ثابت به مشترک خود سرکشی می‌کند ولیکن فواصل زمانی سرکشی به مشترک، قطعی و منظم نیست. یک مشترک که از نرخ ثابت بهره می‌گیرد می‌تواند یک بیت خاصّ را در یکی از فریمهای خود تنظیم کرده و تقاضای سرکشی بدهد تا بتواند ترافیک اضافی (با نرخ متغیر) ارسال نماید.

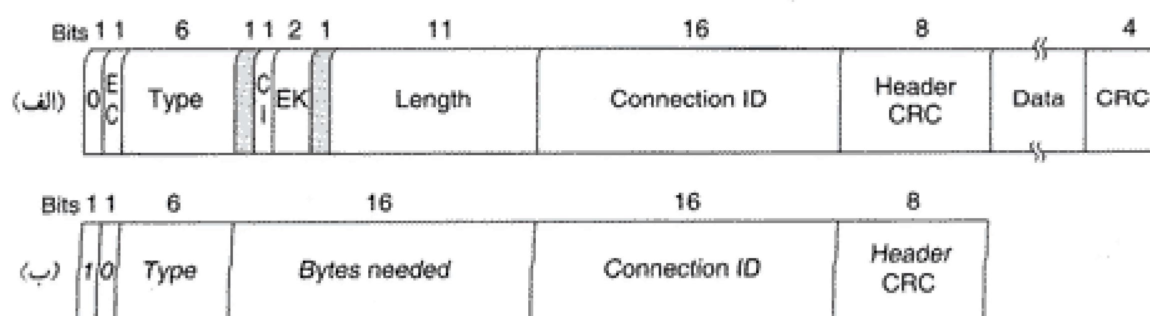
اگر ایستگاهی k بار متوالی سرکشی شود و پاسخی ندهد، ایستگاه ثابت او را در یک «گروه چندپخشی» (Multicast Group) قرار داده و از آن به بعد بصورت اختصاصی سرکشی نخواهد شد. در عوض وقتی به یک «گروه چندپخشی» سرکشی می‌شود هر کدام از ایستگاه‌های گروه می‌توانند پاسخ بدهند و برای دریافت خدمات رقابت کنند. بدین ترتیب ایستگاه‌هائی که ترافیک ناچیزی دارند زمان باارزش سرکشی را هدر نخواهند داد.

خدمات مبتنی بر **«بهترین تلاش»** در موارد متفاوتی کاربرد دارد. در اینجا هیچ سرکشی انجام نمی‌گیرد و هر مشترک برای دریافت این خدمات باید با مشترکین دیگر (که آنها نیز در پی خدمات مبتنی بر بهترین تلاش هستند) رقابت کند. تقاضای پهنای باند با علامتگذاری در یکی از برشهای زمانی ترافیک روبه‌بالا که برای رقابت تدارک دیده شده انجام می‌گیرد. اگر تقاضا به صورت موفقیت‌آمیز اعلان شود، این موفقیت در نگاشت ترافیک روبه‌پائین [فریم مربوطه به Downstream] مشخص خواهد شد. اگر تقاضا موفق نبود، مشترک بایستی مجدداً تلاش کند. برای آنکه تصادمها حداقل شود از الگوریتم عقب‌گرد نمایی در اترنت بهره گرفته شده است.

استاندارد 802.16 ، دو نوع تخصیص پهنای باند تدارک دیده است: تخصیص پهنای باند به ازای هر ایستگاه و تخصیص پهنای باند به ازای هر اتصال (per-connection) . در روش اول، ایستگاه نصب شده در یک ساختمان، کلیهٔ تقاضاهای کاربران را به صورت یکجا جمع کرده و به نیابت از همهٔ آنها تقاضای پهنای باند می‌کند و اگر توانست پهنای باند درخواستی را بدست بیاورد، این پهنای باند را به تناسب بین کاربران تقسیم می‌کند. در روش دوّم، ایستگاه ثابت هر اتصال را مستقلاً مدیریت می‌کند.

۴-۵-۵ ساختار فریم در 802.16

تمام فریمهای MAC با یک سرآیند عمومی شروع می‌شوند. به نحوی که در شکل ۴-۳۴ می‌بینید پس از سرآیند، بخشی اختیاری جهت حمل داده و کد اختیاری کشف خطا (CRC) قرار گرفته است. در فریمهای کنترلی به بخش داده (Payload) نیازی نیست (مثلاً در فریمهایی که تقاضای برش زمانی می‌دهند). کد کشف خطا (Checksum) نیز اختیاری است زیرا در لایهٔ فیزیکی، عملیات تصحیح خطا انجام می‌گیرد و همچنین قرار نیست فریمهای بی‌درنگ *[در صورت خراب شدن] از نو ارسال شوند. اگر قرار نباشد که فریمی از نو ارسال شود چرا با افزودن کد* کشف خطا، خود را به زحمت بیندازیم.



شکل ۴-۳۴. (الف) قالب عمومی فریم (ب) فریم تقاضای پهنای باند.

فیلدهای سرآیند شکل ۴-۳۴-الف را به اختصار معرفی می‌کنیم: بیت EC مشخص می‌کند که آیا بخش داده، رمزنگاری شده است؟ فیلد Type ، نوع فریم را تعیین می‌کند، بالاخص آنکه آیا داده‌ها به صورت مجموعه‌ای از قطعات یا آنکه به صورت یکجا ارسال می‌شود. فیلد CI مشخص می‌کند که آیا فیلد کشف خطا (Checksum) در پایان فریم وجود دارد یا خیر. فیلد EK مشخص‌کننده آنست که کدامیک از کلیدهای رمزنگاری به کارگرفته شده است. (بشرط آنکه رمزنگاری انجام و چندین کلید تعریف شده باشد). فیلد Length طول کلّ فریم را با احتساب سرآیند تعیین می‌کند. فیلد Connection Identifier مشخص می‌کند که فریم متعلق به کدام «اتصال» است. در آخر، فیلد Header CRC در برگیرندهٔ کد کشف خطائی است که فقط از بخش سرآیند و با استفاده از چندجمله‌ای x^8+x^2+x+1 استخراج می‌شود.

نوع دوّم سرآیند که فقط برای فریمهائی که پهنای باند تقاضا می‌دهند تعریف شده، در شکل ۴-۳۴-ب نشان داده شده است. این سرآیند به جای بیت صفر با بیت ۱ شروع شده ولیکن ترکیب کلّی آن با ترکیب فریم عمومی [شکل ۴-۳۴-الف] مشابه است، با این تفاوت که بایت دوّم و سوّم این فریم یک عدد ۱۶ بیتی را تشکیل داده و محتوای آن مشخص می‌کند که برای مبادلهٔ تعداد مشخص بایت، به چه اندازه پهنای باند نیاز است. فریم تقاضای پهنای باند دارای بخش حمل داده (Payload) و کد کشف خطا برای کلّ فریم نیست.

اگر چه می‌توان مفصلاً در خصوص 802.16 شرح داد ولی در اینجا به بحث خاتمه می‌دهیم. برای آگاهی بیشتر مستقیماً به استاندارد آن مراجعه کنید.

۶-۴ بلوتوث (Bluetooth)

در سال ۱۹۹۸ شرکت ال.ام. اریکسون علاقمند شد تا گوشی تلفنهای همراه تولیدی او بتوانند بصورت بی‌سیم به ابزارهای دیگر (مثل PDA) وصل شوند. اریکسون و چهار شرکت دیگر (آی‌بی‌ام، اینتل، نوکیا و توشیبا) یک «گروه SIG»[1] تشکیل دادند تا استانداردی بی‌سیم برای اتصال ابزارهای مخابراتی/رایانه‌ای و ابزارهای جانبی آنها طراحی کنند که بردی کوتاه، مصرف توان پائین و قیمتی ارزان داشته باشد. نام این پروژه، بلوتوث انتخاب شد که برگرفته از نام «هرالد بلاتاند دوّم» (مشهور به Bluetooth) ، یکی از پادشاهان وایکینگ است (۹۸۱_۹۴۰) که دانمارک و نروژ را با هم متحد کرد (البته با زور و بدون کابل!!!)

اگر چه تفکر اصلی، رهایی از شرّ کابلهای مابین دستگاه‌های دیجیتالی بود ولی به سرعت در حوزه‌های دیگر نیز گسترش یافت و به تدریج در حیطهٔ شبکه‌های محلی بی‌سیم نیز وارد شد. اگرچه گسترش و رشد این استاندارد، روز بروز کاربرد آنرا بیشتر می‌کرد ولی در عوض چالشهائی بین این استاندارد و 802.11 پدید آورد. نقطه شدّت این چالش آنجاست که این دو سیستم از لحاظ الکتریکی با یکدیگر تداخل فرکانسی دارند. اشاره به این نکته مهم است که شرکت هیولت پاکارد چندین سال قبل از آن شبکه‌ای مبتنی بر نور مادون‌قرمز برای وصل بی‌سیم دستگاه‌های جانبی کامپیوتر عرضه کرده بود، ولی استقبال چندانی از آن نشد.

فارغ از همهٔ اینها، در ژولای ۱۹۹۹ گروه طراح بلوتوث، مشخصات هزار و پانصد صفحه‌ای از نسخهٔ ۱ آن یعنی V1.0 را منتشر نمود. به فاصلهٔ کوتاهی، گروه استانداردسازی IEEE که در اندیشه تدوین استاندارد 802.15 برای «شبکه‌های شخصی بی‌سیم» بودند مستندات استاندارد بلوتوث را به عنوان مبنای کار خود برگزیدند و شروع به پالایش و تکمیل آن نمودند. اگر چه استانداردسازی چیزی که مشخصات تفصیلی و مشروح آن در اختیار است و پیاده‌سازیهای متنوع و ناسازگار ندارد (که نیاز به یکنواخت‌سازی و هماهنگی داشته باشد) عجیب به نظر می‌رسد

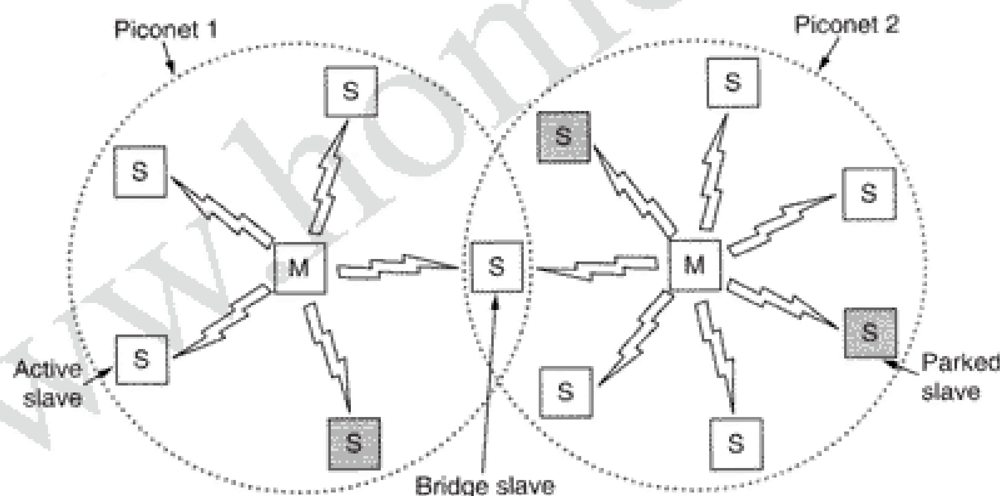
۱. SIG به معنای گروهی با گرایش خاصّ یا همان کنسرسیوم است.

ولی تاریخ نشان داده که وجود یک «استاندارد باز» که توسط سازمانی بی‌طرف مثل IEEE تدوین و مدیریت می‌شود عموماً کاربری یک تکنولوژی را ترویج و ترغیب خواهدکرد. اگر بخواهیم اندکی دقیقتر سخن بگوئیم باید اشاره کنیم که توصیف استاندارد بلوتوث برای سیستمی کامل تدوین شده که از لایهٔ فیزیکی تا لایهٔ کاربرد را در بر می‌گیرد درحالیکه کمیته IEEE 802.15 فقط لایه‌های فیزیکی و پیوند داده را استانداردسازی کرده و باقیماندهٔ پشتهٔ پروتکلی خارج از برنامه این استاندارد است.

هرچند IEEE اولین «**استاندارد شبکهٔ شخصی**»[1] (PAN) را در سال ۲۰۰۲ با عنوان 802.15.1 به تصویب رساند ولی هنوز کنسرسیوم بلوتوث فعال و سرگرم بهبود و توسعهٔ آنست. اگر چه نسخهٔ استاندارد عرضه شده توسط کنسرسیوم بلوتوث و IEEE یکی نیستند ولی انتظار می‌رود بزودی به یک استاندارد واحد همگرا شوند.

۴ـ۶ـ۱ معماری بلوتوث

اجازه بدهید بررسی سیستم بلوتوث را با مروری سریع بر دستاوردها و اهداف آن آغاز نمائیم. واحد پایه در سیستم بلوتوث یک «**پیکونت**» (Piconet) است که از یک «گره اصلی» (Master Node) و حداکثر هفت «گره پیرو فعّال» (Active Slave Node) به فاصلهٔ حداکثر ده‌متر ، تشکیل شده است. در یک فضای بزرگ و واحد می‌توان چندین پیکونت داشت و حتی می‌توان آنها را از طریق یک گره که نقش پل (Bride) ایفاء می‌کند به هم متصل کرد (به شکل ۴ـ۳۵ نگاه کنید). به مجموعه‌ای از پیکونتهای متصل بهم اصطلاحاً Scatternet (شبکه متفرق/پراکنده) گفته می‌شود.



شکل ۴ـ۳۵. دو پیکونت می‌توانند با اتصال بهم یک Scatternet تشکیل بدهند.

در یک پیکونت علاوه بر هفت گره فعال پیرو، می‌تواند تا ۲۵۵ گره غیرفعال وجود داشته باشد. اینها دستگاه‌هائی هستند که گره اصلی آنها را در حالت استراحت و کم‌توان وارد کرده تا مصرف باطری آن کاهش یابد. یک ایستگاه در حالت غیر فعال هیچ کاری نمی‌تواند انجام بدهد به جز آنکه به سیگنال فعال‌سازی خود یا سیگنال Beacon که از گره اصلی می‌رسد، پاسخ بدهد. به غیر از این حالات، دو حالت میانی در مصرف توان به نامهای حالت Hold و Sniff نیز وجود دارد که در اینجا بدان نخواهیم پرداخت.

دلیل طراحی Master/Slave (اصلی/پیرو) آن بود که طراحان آن در نظر داشتند قیمت کل سیستم بلوتوث پیاده‌سازی شده برروی تراشه، زیر پنج دلار باشد. نتیجهٔ این تصمیم‌گیری آنست که گره‌های پیرو [مثل صفحه

۱. Personal Area Network

کلیدها، موس، چاپگر] تقریباً غیر هوشمند و ساده هستند و اساساً آنچه را که گره اصلی (Master) به آنها دستور بدهد اجرا می‌کنند. یک پیکونت سیستمی مبتنی بر TDM متمرکز (Centralized TDM) است که در آن هستهٔ مرکزی (یعنی گره اصلی یا Master) بر سیگنال ساعت نظارت دارد و تعیین می‌کند که چه دستگاهی و در کدام برش زمانی (Slot) مخابره داشته باشد. تبادل اطلاعات صرفاً بین گره مرکزی و گره‌های پیرو انجام می‌شود و ارتباط مستقیم دو گره پیرو [مثلاً دو صفحه کلید یا دو چاپگر] ممکن نیست.

۴-۶-۲ کاربردهای بلوتوث

بیشتر پروتکلهای شبکه فقط کانالی را بین چند مولفهٔ مخابراتی ، سازماندهی و ایجاد می‌کنند و اجازه می‌دهند طراحان برنامه‌های کاربردی به پیاده‌سازی هر آنچه که مورد نیاز است بپردازند. به عنوان مثال 802.11 مشخص نکرده که کاربران باید صرفاً از کامپیوتر کیفی خود برای خواندن ایمیل یا جستجو در وب یا هر چیز دیگری بهره بگیرند. برعکس، در تشریح بلوتوث نسخهٔ V1.1 از ۱۳ کاربرد مختلف که باید از آنها پشتیبانی شود، نام برده شده و برای هر یک، پشتهٔ پروتکلی متفاوتی ارائه گردیده است. متاسفانه این راهکار به پیچیدگی بسیار زیاد منتهی می‌شود و ما از آن صرف نظر خواهیم کرد. این سیزده کاربرد که «**پروفایل**» نام گرفته‌اند در شکل ۴-۳۶ فهرست شده‌اند. با نگاهی اجمالی به پروفایلها ممکن است به آنچه که کنسرسیوم بلوتوث در پی انجام آن بوده بیشتر پی ببریم.

نام پروفایل	عملکرد
Generic access	پروسیجرهایی برای مدیریت لینک
Service discovery	پروتکلی برای کشف سرویسهای عرضه شده
Serial port	جایگزینی برای کابل معمولی پورت سریال
Generic object exchange	مدل ارتباطی بین سرویس‌دهنده و مشتری برای جابجایی (انتقال) اشیا را تعریف می‌کند.
LAN access	پروتکل ارتباطی بین کامپیوتر همراه و شبکه محلی ثابت (با کابل سیمی)
Dial-up networking	امکان برقراری تماس یک کامپیوتر کیفی را از طریق تلفن همراه فراهم می‌آورد.
Fax	امکان ارتباط بین یک دستگاه دورنگار بی‌سیم و تلفن همراه را فراهم می‌آورد.
Cordless telephony	ارتباط بین یک دستگاه گوشی تلفن بی‌سیم و ایستگاه ثابت و محلی آن را برقرار می‌کند.
Intercom	امکاناتی برای واکی تاکی دیجیتال
Headset	امکان ارتباط از طریق هندزفری (Handsfree) را فراهم می‌آورد.
Object push	روشی برای مبادله اشیا ساده
File transfer	عرضه کننده امکانات عمومی بیشتر جهت انتقال فایل
Synchronization	امکان سنکرون‌سازی داده‌های یک PDA با کامپیوتری دیگر را فراهم می‌آورد.

شکل ۴-۳۶. پروفایلهای بلوتوث.

«**پروفایل عمومی دسترسی**» (Generic Access) حقیقتاً یک برنامهٔ کاربردی نیست بلکه بیشتر یک زیربناست که بر اساس آن برنامه‌های کاربردی حقیقی ساخته و پیاده می‌شوند. وظیفهٔ اصلی آن ارائه تمهیداتی است که بتوان بین گره اصلی (Master) و گره‌های پیرو (Slave) یک کانال مطمئن برقرار و آنرا حفظ کرد. «**پروفایل تشخیص خدمات**» (Service Discovery) که آنهم تقریباً عمومی و کلّی است توسط دستگاه‌ها برای آگاهی از خدماتی که دیگر دستگاه‌ها ارائه می‌دهند، استفاده می‌شود. تمام دستگاه‌های مبتنی بر بلوتوث موظف به پیاده‌سازی این دو پروفایل هستند. بقیه پروفایلها اختیاریند.

«**پروفایل درگاه سریال**» (Serial Port) یک پروتکل انتقال است که بقیه پروفایلها از آن بهره می‌گیرند. این

پروفایل یک درگاه سریال را شبیه‌سازی می‌کند و بطور خاص برای کاربردهای قدیمی که به خط سریال نیاز دارند، سودمند است.

«**پروفایل عمومی مبادلهٔ شیء**» (Generic Object Exchange) یک ارتباط مبتنی بر مدل مشتری/سرویس‌دهنده ، برای انتقال داده‌ها تعریف کرده است. اگرچه همیشه مشتری، آغازکنندهٔ عملیات است ولیکن یک گره پیرو می‌تواند هم سرویس‌دهنده و هم مشتری باشد. همانند پروفایل «درگاه سریال» این پروفایل نیز زیربنای دیگر پروفایلهاست.

گروه سه‌تائی پروفایلهای بعدی به منظور کاربردهای شبکه‌ای (Networking) تعریف شده‌اند. «**پروفایل دسترسی به LAN**» اجازه می‌دهد که یک دستگاه مبتنی بر بلوتوث به یک شبکهٔ ثابت متصل شود. این پروفایل رقیب مستقیم 802.11 است. «**پروفایل شبکهٔ مبتنی بر شماره‌گیری**» (Dialup) انگیزهٔ اصلی کل این پروژه بوده است. این پروفایل اجازه می‌دهد که یک کامپیوتر کیفی بتواند به یک تلفن همراه که دارای مودم داخلی بی سیم است متصل شود. «**پروفایل دورنگار**» (Fax) شبیه به پروفایل شماره‌گیری است با این تفاوت که اجازه می‌دهد ماشینهای دورنگار بی‌سیم از طریق یک دستگاه تلفن همراه و بدون نیاز به سیم، اقدام به ارسال یا دریافت دورنگار کنند.

سه پروفایل بعدی در خصوص تلفن کاربرد دارند. پروفایل «**تلفن بی‌سیم**» (Cordless Telephony) راهی را برای اتصال گوشی یک تلفن بی‌سیم به ایستگاه ثابت است. در حال حاضر تلفنهای بی‌سیم خانگی را نمی‌توان به عنوان تلفن همراه به کارگرفت ولی در آینده شاید تلفن بی‌سیم و تلفن همراه در هم ادغام شوند. «**پروفایل Intercom**» این امکان را فراهم می‌کند تا دو تلفن ، شبیه به «واکی‌تاکی» (Walkie/Talkie) بهم متصل گردند. نهایتاً «**پروفایل گوشی** –Headset–» امکان ارتباط بی‌سیم بین گوشی و ایستگاه ثابت (مثل هندزفری –Hands Free-) را فراهم می‌کند که به عنوان مثال برای صحبت با تلفن در حین رانندگی مفید است.

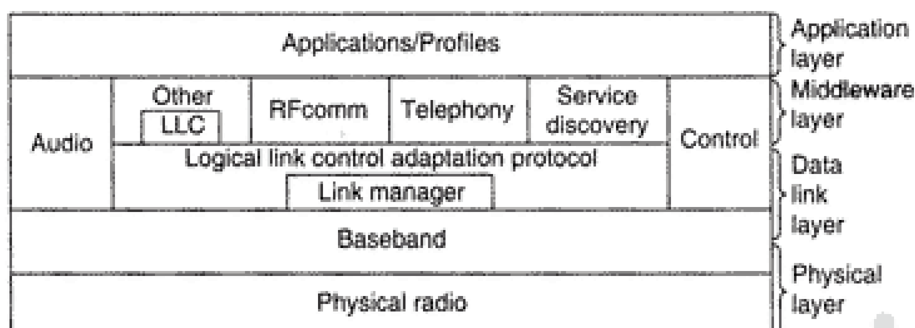
سه پروفایل باقیمانده برای مبادلهٔ اشیاء بین دو ابزار تعریف شده است. اشیاء می‌توانند فایلهای داده، تصویر یا کارتهای تجاری باشند. «**پروفایل سنکرواسیون**» ، برای بار کردن داده در درون کامپیوتر کیفی یا PDA (کامپیوترهای دستی) در حین ترک منزل و جمع آوری اطلاعات پس از برگشت، مفید است.

آیا واقعاً نیاز بوده که تمام این کاربردها به تفصیل تحلیل شوند و برای هرکدام پشته پروتکلی متفاوتی تعریف گردد؟ شاید نه! امّا بخشهای متفاوت این استاندارد را گروه‌های کاری مختلف طراحی کرده‌اند و چون هر یک از این گروه‌ها برروی نیازهای خاص خود متمرکز بودند، در نتیجه هر یک پروفایل موردنظر خود را پدید آوردند. برای توجیه این موضوع به قانون کانوی (Conway's Law) بیندیشید. (در یکی از شماره‌های مجلهٔ Datamation در آوریل ۱۹۶۸ ، ملوین کانوی مشاهدات خود را بدین نحو منتشر کرد که اگر n شخص را به نوشتن یک کامپایلر بگمارید، آنچه که بدست خواهید آورد یک کامپایر n–pass است [یعنی کامپایلری که در آن تعداد مراحل ترجمهٔ یک برنامه ، n گذر می‌باشد.] بعبارت عام ساختار نهائی یک نرم‌افزار آینهٔ تمام‌نمای ترکیب گروهی است که آن را تولید کرده‌اند. شاید می‌شد که به جای سیزده پشته پروتکلی به دو پشتهٔ کلّی بسنده کرد: یکی برای انتقال فایل و دیگری برای انتقال بی‌درنگ جریان اطلاعات.

۳-۶-٤ پشتهٔ پروتکلی بلوتوث

استاندارد بلوتوث پروتکلهای متعددی دارد که بطور ناموزون در چند لایه گروه‌بندی شده‌اند. ساختار لایه‌ها از مدل OSI ، مدل TCP/IP ، مدل 802 یا هر مدل شناخته شدهٔ دیگر تبعیت نمی‌کند. با این وجود IEEE در حال اصلاح بلوتوث است تا با مدل 802 سازگارتر شود. معماری پروتکل بلوتوث که توسط کمیته 802 اصلاح شده ، در شکل ۴-۳۷ مشاهده می‌شود.

لایهٔ زیرین، «**لایهٔ رادیو فیزیکی**» است که تقریباً متناظر با لایهٔ فیزیکی از مدل OSI یا مدل 802 می‌باشد. این



شکل ۴-۳۷. نسخهٔ 802.15 از معماری پروتکل بلوتوث.

لایهٔ با انتقال رادیوئی و مدولاسیون سرکار دارد. بسیاری از ملاحظاتی که در طراحی این لایه باید مورد توجه قرار می‌گرفت آن بود که سیستم ارزان قیمت باشد و بطور انبوه در بازار عرضه شود.

«لایهٔ باند پایه» (Baseband) از جهاتی شبیه به زیرلایهٔ MAC است ولیکن مولفه‌هائی از لایهٔ فیزیکی را نیز در بر می‌گیرد. این لایه با مسائلی مثل چگونگی نظارت گره اصلی (Master) بر برشهای زمانی و چگونگی گروه‌بندی این برشهای زمانی در قالب فریمها، سرکار دارد.

سپس لایه‌ای شامل یک گروه از پروتکلهای مرتبط با هم، تعریف شده است. مدیر لینک (Link Manager) عملیات ایجاد کانالهای منطقی بین دستگاه‌ها، شامل «مدیریت توان مصرفی»، «احراز هویت» (Authentication) و «کیفیت خدمات» (QoS) را بر عهده دارد. «پروتکل تطبیق کنترل لینک منطقی» (که اغلب L2CAP گفته می‌شود) وظیفه دارد لایه‌های بالایی را از درگیری با جزئیات ارسال، راحت کند. این لایهٔ مشابه با استاندارد زیرلایهٔ 802 LLC است ولی از لحاظ مسائل فنّی با آن متفاوت است. دو پروتکل «کنترل» و «صدا» همانگونه که از نامشان بر می‌آید با مسائل انتقال صدا و عملیات کنترل سروکار دارند. برنامه‌های کاربردی می‌توانند بدون نیاز به L2CAP، مستقیماً این دو پروتکل را به خدمت بگیرند.

لایهٔ بعدی یک لایهٔ میانی است (Middleware) و تلفیقی از پروتکلهای متفاوت را در برمی گیرد. در این لایه از پروتکل IEEE 802 LLC، بمنظور سازگاری با دیگر شبکه‌های سری 802 استفاده شده است. پروتکلهای RFcomm، Telephony و Service Discovery صرفاً مرتبط با بلوتوث هستند. RFcomm (Radio Frequency Communication)، پروتکلی جهت شبیه‌سازی استاندارد درگاه سریال (Serial port) است که در تمام PCها از آن برای اتصال صفحه کلید، موس، مودم و امثال آن استفاده می‌شود. این پروتکل برای آن طراحی شده تا بتوان از دستگاه‌های قدیمی بسهولت استفاده کرد. «پروتکل تلفنی» (Telephony) پروتکلی بی‌درنگ است که برای سه پروفایل مبتنی بر انتقال صدا بکار می‌آید. این پروتکل همچنین تنظیم و قطع ارتباط را بر عهده دارد. نهایتاً پروتکل «تشخیص خدمات» (Service Discovery) برای کشف و تشخیص انواع خدماتی که درون شبکه عرضه می‌شود، کاربرد دارد.

بالاترین لایه، محّل قرار گرفتن انواع برنامه‌های کاربردی و پروفایلها است. این لایه برای انجام کار از خدمات پروتکلهای موجود در لایه‌های زیر بهره می‌گیرد. هر برنامهٔ کاربردی، زیرمجموعه‌ای از پروتکلهای مختص به خود را به خدمت می‌گیرد. ابزارهای ویژه‌ای مثل گوشی بی‌سیم (Headset) بسته به نوع برنامهٔ کاربردی آنها، فقط به برخی از پروتکل‌ها نیازمندند.

در بخشهای بعدی سه لایهٔ پائینی از پشتهٔ پروتکلی بلوتوث را بررسی خواهیم کرد چرا که تقریباً متناظر با زیرلایه‌های فیزیکی و MAC است.

۴-۶-۴ لایهٔ رادیوئی در بلوتوث

لایهٔ رادیوئی بیتها را از گره اصلی به گره پیرو و بالعکس، منتقل می‌کند. این لایه، سیستمی با توان کم و برد ده متر است که در باند فرکانسی 2.4GHz ISM عمل می‌کند. این باند به ۷۹ کانال یک مگاهرتزی تقسیم می‌شود. مدولاسیون به کاررفته FSK (Frequency Shift Keying) و هر هرتز (هر سیکل) معادل یک بیت است که جمعاً نرخ یک مگابیت بر ثانیه را در اختیار می‌گذارد ولی بیشتر این پهنای باند به دلیل سربار تلف می‌شود. برای تخصیص مناسب این کانالها از روش پرش فرکانس در طیف گسترده (Spread Spectrum) با نرخ پرش 1600 hops/sec و Dwell time معادل ۶۲۵ میکروثانیه بهره گرفته شده است.

چون بلوتوث و 802.11، هر دو در باند 2.4GHz ISM و دقیقاً در همان ۷۹ کانال کار می‌کنند لذا با یکدیگر تداخل فرکانسی خواهند داشت. از آنجایی که پرش فرکانس در بلوتوث سریعتر از 802.11 است لذا دستگاه‌های مبتنی بر بلوتوث به احتمال بیشتری در انتقال 802.11 اخلال خواهد کرد تا 802.11 در بلوتوث! چون 802.11 و 802.15 هر دو استانداردهای IEEE هستند لذا IEEE به دنبال راه حلّی برای این مشکل می‌گردد ولی حَل این مشکل چندان ساده نیست چرا که هر دو سیستم، بدلیل مشابهی از این باند فرکانسی بهره گرفته‌اند: زیرا، برای استفاده از این باند فرکانسی، به اخذ هیچ مجوزی نیاز نیست. استاندارد 802.11a از باند دیگر ISM (باند 5GHz) استفاده می‌کند ولیکن برد کمتری نسبت به 802.11b دارد (بدلیل ماهیت فیزیکی امواج رادیوئی این باند) لذا استفاده از 802.11a راه حل مناسبی نخواهد بود. برخی از شرکتها این مشکل را با ممنوعیت استفاده از بلوتوث حل کرده‌اند. راه حل بازاری این مشکل آنست که صبر کنیم تا عاقبت شبکه‌ای که از لحاظ اقتصادی و سیاسی جایگاه مستحکم‌تری در بازار پیدا کرد، از طرف مقابل بخواهد تا استاندارد خود را برای حَل مشکل تداخل، اصلاح نماید. در این خصوص مطالبی در مرجع (Lansford et al. ,2001) ارائه شده است.

۵-۶-۴ لایهٔ باند پایه در بلوتوث

لایهٔ باند پایه شبیه‌ترین بخش بلوتوث با زیرلایهٔ MAC است. این لایه، دنبالهٔ بیتهای خام را به فریمها تبدیل می‌کند و بدین منظور چندین قالب مهم فریم تعریف نموده است. در ساده‌ترین حالت، گره اصلی در هر **پیکونت** دنباله‌ای از برشهای زمانی ۶۲۵میکروثانیه‌ای (Time Slot) تولید می‌کند، با این توصیف که ارسال داده‌های گره اصلی در برشهای زمانی با شمارهٔ زوج انجام می‌شود و گره‌های پیرو (Slaves) در برشهای زمانی فرد ارسال می‌نمایند. این روش مشابه با روش تسهیم زمانی (TDM) معمولی است که در آن، گره اصلی نیمی از برشهای زمانی را در اختیار دارد و بقیه گره‌ها (حداکثر هفت گره) در نیم دیگر سهیم هستند. ارسال هر فریم می‌تواند ۱، ۳ یا ۵ برش زمانی طول بکشد.

در هر پرش فرکانسی ۲۵۰ تا ۲۶۰ میکروثانیه طول خواهد کشید تا مدار رادیوئی بتواند پایدار شود. پایداری سریعتر نیز ممکن است ولی هزینهٔ پیاده‌سازی بیشتری دارد. برای فریمهای که فقط به یک برش زمانی نیاز دارند پس از هر پرش فرکانسی، ۳۶۶ بیت از کل ۶۲۵ بیت باقی خواهد ماند. از این مقدار ۱۲۶ بیت به **«کد دسترسی»** (Access Code) و **«سرآیند»** اختصاص دارد و ۲۴۰ بیت، برای داده‌ها باقی می‌ماند. وقتی پنج برش زمانی به هم ملحق می‌شوند [برای ارسال فریمهائی به طول ۵ برش] تنها به یک زمان پایداری نیاز خواهد بود و طبعاً زمان کوتاهتری برای زمان پایداری، تلف می‌شود و ۳۱۲۵=۶۲۵×۵ بیت در پنج برش زمانی ارسال می‌گردد که از این مقدار ۲۷۸۱ بیت برای ارسال داده در اختیار **«لایهٔ باند پایه»** خواهد بود. بنابراین در بلوتوث فریمهای طولانی کارآمدتر از فریمهای کوچک هستند.

هر فریم برروی یک کانال منطقی که اصطلاحاً «لینک بین گره اصلی و گره پیرو» نام دارد، ارسال خواهد شد. دو نوع لینک وجود دارد: لینک اول ACL است (Asynchronous Connection-Less) که برای ارسال داده‌ها در

برشهای زمانی نامنظم کاربرد دارد. این داده‌ها از لایهٔ L2CAP در سمت فرستنده تولید و در سمت گیرنده تحویل لایهٔ L2CAP می‌شوند. ترافیک داده‌های ACL مبتنی بر روش «**بیشترین تلاش**» (Best Effort) ارسال می‌شود ولی هیچ تضمینی در تحویل آنها نیست. فریمها می‌توانند گم شده یا از بین بروند، بدون آنکه ارسال مجدد شوند. هر گره پیرو فقط می‌تواند یک لینک ACL با گره اصلی داشته باشد.

لینک دیگر، SCO (Synchronous Connection Oriented) نام دارد که برای ارسال داده‌های بی‌درنگ، مثل ارتباط تلفنی کاربرد دارد. این نوع کانال با تخصیص برشهای زمانی مشخص در هر دو جهت، ایجاد می‌شود. بدلیل آنکه لینکهای SCO نسبت به زمان حساس هستند فلذا فریمهای ارسالی برروی این لینک هرگز ارسال مجدداً (Retransmit) نخواهند شد، در عوض از روش «تصحیح مستقیم خطا» استفاده شده تا اطمینان بیشتری داشته باشد.[ارسال مجدد فریمهای خراب بر عهدهٔ لایهٔ بعدی است و در این لایه –در صورت امکان– به تصحیح خطا بسنده می‌شود. –م] هر گره پیرو می‌تواند حداکثر سه لینک SCO با گره اصلی داشته باشد. هر لینک SCO می‌تواند یک کانال صدا مبتنی بر PCM با نرخ 64000 بیت برثانیه را حمل کند.

۴ـ۶ـ۶ لایهٔ L2CAP در بلوتوث

لایهٔ L2CAP سه دسته عملیات مهم را بر عهده دارد: اول آنکه بسته‌هائی با طول حداکثر ۶۴کیلوبایت را از لایه‌های بالائی پذیرفته و آنها را جهت انتقال، به فریمهای کوچکتری می‌شکند. در سمت مقابل این فریمها مجدداً به بستهٔ اصلی بازسازی خواهند شد.

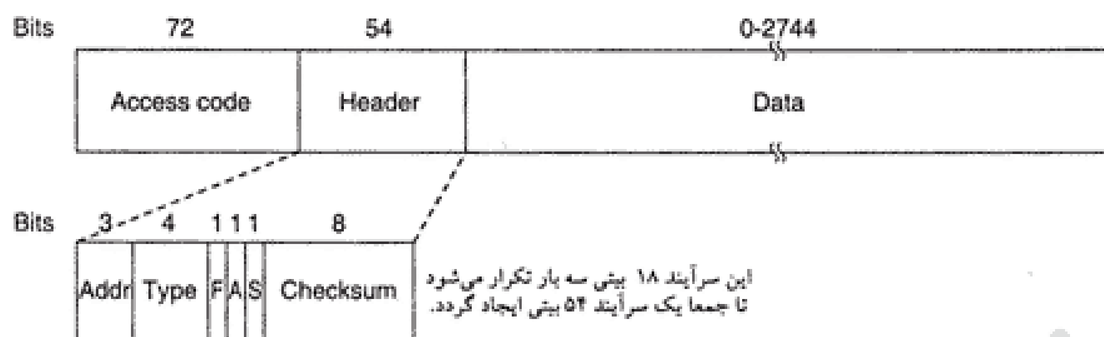
دوّم آنکه این لایهٔ عمل جمع‌آوری و توزیع بسته‌هائی که از چندین مبداء آمده [یا به چندین مقصد می‌روند] را برعهده دارد. وقتی یک بسته بازسازی می‌شود، لایهٔ L2CAP تعیین خواهد کرد که باید به کدام پروتکل در لایهٔ بالاتر (مثلاً RFcomm یا Telephony) تحویل شود.

سوم آنکه این لایه، عملیات تامین «کیفیت خدمات» (QoS) را بر عهده دارد، چه در هنگام ایجاد لینک و چه در خلال عملکرد طبیعی. همچنین در زمان ایجاد لینک، بر سر اندازهٔ حداکثر و مُجاز طول داده، توافق صورت می‌گیرد تا ابزارهائی با طول بسته بزرگ از ارسال چنین بسته‌ای به ابزارهائی با طول بستهٔ کوچک اجتناب کنند. از آنجایی که تمام ابزارهای مبتنی بر بلوتوث نمی‌توانند بسته‌هائی با طول ۶۴کیلوبایت را بپذیرند فلذا به این ویژگی نیاز است.

۴ـ۶ـ۷ ساختار فریم در بلوتوث

چندین نوع قالب فریم در بلوتوث وجود دارد که مهمترین آنها در شکل ۴–۳۸ نشان داده شده است. این فریم با فیلد «کد دسترسی» (Access Code) شروع می‌شود که عموماً هویت یک گره اصلی (Master) را مشخص خواهد کرد تا بدینگونه یک گره پیرو که در برد رادیوئی دو گره اصلی قرار دارد، گیرندهٔ حقیقی ترافیک داده‌ها را مشخص نماید. سپس یک سرآیند ۵۴بیتی آمده که شامل فیلدهای معمولی زیرلایهٔ MAC است. سپس فیلد داده قرار گرفته که حداکثر ۲۷۴۴ بیت را (برای انتقال در پنج برش زمانی) در بر می‌گیرد. در فریمهائی که تنها در یک برش زمانی ارسال می‌شوند، قالب فریم همین است با این تفاوت که فیلد دادهٔ آنها حداکثر ۲۴۰ بیت است.

حال اجازه بدهید به فیلدهای سرآیند، نگاهی سریع بیندازیم. «فیلد آدرس» هویت گیرندهٔ فریم را از بین هشت دستگاه فعال در هر پیکونت مشخص می‌کند. «فیلد Type»، اولاً نوع فریم را (از بین انواع ACL، SCO، POLL یا NULL)، ثانیاً روش تصحیح خطای داده‌ها را و ثالثاً تعداد برشهای زمان که ارسال فریم جاری بدان نیاز دارد را مشخص می‌نماید. «بیت Flow» توسط گره‌های پیرو و زمانی تنظیم [فعال] می‌شود که بافر آنها پر شده باشد و نتوانند دادهٔ بیشتری دریافت کنند. این بیت، شکل ابتدائی کنترل جریان داده‌ها، به حساب می‌آید. «بیت



شکل ۴ـ۳۸. قالب کلّی فریم داده در بلوتوث.

Acknowledgement» بدین منظور است تا دریافت صحیح یک فریم از طرف مقابل ، در فریم ارسالی جاسازی و اعلام شود [یعنی فرآیند Piggybacking]. «بیت Sequence» برای شماره‌گذاری فریمهاست تا بسته‌های تکراری کشف شوند. در بلوتوث پروتکل ارسال مجدّد، روش «توقف و انتظار» (Stop & wait) است و طبعاً یک بیت برای شماره‌گذاری فریمها کفایت می‌کند. در ادامه «فیلد هشت بیتی Checksum» برای کشف خطای احتمالی در سرآیند تعریف شده است. کلّ این سرآیند ۱۸ بیتی سه بار تکرار می‌شود تا سرآیند ۵۴ بیتی نشان داده شده در شکل ۴–۳۸بوجود آید. در سمت گیرنده، با یک مدار ساده سه نسخهٔ تکراری هر بیت بررسی می‌شود. اگر هر سه بیت مثل هم بودند، آن بیت پذیرفته می‌شود در غیر اینصورت ، بیتی که بیشترین تکرار را دارد قبول می‌شود. بدین ترتیب، برای ارسال یک سرآیند ۱۰ بیتی ظرفیتی معادل ۵۴بیت صرف می‌شود. دلیل آن این بوده که برای ارسال مطمئن داده‌ها در محیطی سرشار از نویز و با استفاده از ابزاری ارزان قیمت و توانی ناچیز (2.5mW) و قدرت پردازش پائین، به افزونگی بسیار زیادی نیاز خواهد بود.

برای فیلد داده در فریمهای ACL ، قالبهای متفاوتی تعریف شده است. فریمهای SCO ساده‌تر هستند: فیلد داده همیشه ۲۴۰ بیتی است. سه گزینهٔ دیگر نیز تعریف شده که در آنها مقدار واقعی داده‌ها ۸۰، ۱۶۰ یا ۲۴۰ بیتی است و باقیماندهٔ بیتها برای تصحیح خطا به کار می‌آیند. در مطمئن‌ترین نسخه (یعنی دادهٔ ۸۰بیتی)، بخش داده همانند سرآیند، سه بار متوالی تکرار می‌شود.

از آنجایی که گره پیرو (Slave) تنها می‌تواند از برشهای زمانی با شمارهٔ فرد استفاده نماید فلذا در هر ثانیه ۸۰۰ برش زمانی بدست خواهد آورد. (همینطور گره اصلی) بدین ترتیب با فیلد دادهٔ ۸۰ بیتی، ظرفیت کانال از سمت گره پیرو به سمت گره اصلی معادل ۶۴۰۰۰ بیت بر ثانیه خواهد بود (ظرفیت کانال از گره اصلی به گره پیرو نیز ۶۴۰۰۰ بیت است) لذا این کانال دقیقاً برای یک کانال صوتی دو طرفهٔ مبتنی بر PCM کافی است. (دلیل انتخاب نرخ 1600 Hops/sec برای تغییر فرکانس همین بوده است). این اعداد و ارقام بدین معنا هستند که یک کانال صوتی دو طرفهٔ PCM با نرخ ۶۴۰۰۰ بیت برثانیه ، در حالت مطمئن [یعنی وقتی داده‌های ۸۰بیتی با سه بار تکرار ارسال می‌شوند]، کلّ پهنای باند موجود در پیکونت را (علیرغم پهنای باند 1 Mbps آن) ، اشباع خواهد کرد. با گزینهٔ نامطمئنتر (یعنی ۲۴۰ بیت در هر برش زمانی بدون هیچگونه افزونگی یا تکرار) می‌توان از حداکثر سه کانال صوتی همزمان حمایت کرد و به همین دلیل حداکثر سه لینک SCO برای هر گره پیرو مجاز شمرده شده است.)

مطالب فراوانتری در خصوص بلوتوث می‌توان گفت که از حوصله این کتاب خارج است. برای آگاهی بیشتر به مراجع زیر مراجعه نمائید.

Bhagwat, 2001; Bisdikian, 2001; Bray and Sturman, 2002; Haartsen, 2000; Johansson et al., 2001; Miller and Bisdikian, 2001; Sairam et al., 2002)

۷-٤ هدایت در سطح لایهٔ پیوند داده‌ها (Data Link Layer Switching)

بسیاری از سازمانها دارای LANهای متعددی هستند و تمایل دارند آنها را به هم متصل کنند. شبکه‌های محلی (LAN) را می‌توان از طریق دستگاه‌هائی که در لایهٔ پیوند داده‌ها عمل می‌کنند و «پل» (Bridge) نامیده می‌شوند به هم متصل نمود. «پلها» برای مسیریابی و هدایت داده‌ها، آدرسهای «لایهٔ پیوند داده‌ها» را بررسی می‌نمایند. از آنجایی که قرار نیست محتوای فیلد داده (از فریمهائی که باید هدایت شوند) بررسی گردد لذا این فریمها می‌توانند بسته‌های IPv4 (که اکنون در اینترنت به کار می‌رود)، IPv6 (که در آینده در اینترنت به کار گرفته خواهد شد)، بسته‌های AppleTalk ، ATM ، OSI یا هر نوع بستهٔ دیگر را در خود حمل کنند. برخلاف پل ، «مسیریابها» آدرس درون بسته‌ها را بررسی کرده و بر این اساس، آنها را هدایت (مسیریابی) می‌کنند. اگرچه این تعریف تمایز بین «مسیریاب» و «پل» را تعیین می‌کند ولی پیشرفتهای جدیدی مثل ابداع شبکهٔ LAN مبتنی بر سوئیچ (Switched Ethernet) ، آب را گل کرده و به نحوی که در بخشهای آتی بدانها خواهیم پرداخت این تمایز را ابهام‌آلود کردند! در بخشهای بعدی به پلها و سوئیچها و بالاخص آنهائی که برای اتصال شبکه‌های محلی 802 به کار می‌آیند، نگاهی خواهیم انداخت. برای بررسی دقیق پلها، سوئیچها و عناوین مهم در این خصوص، به مراجع (Perlman , 2000) مراجعه نمائید.

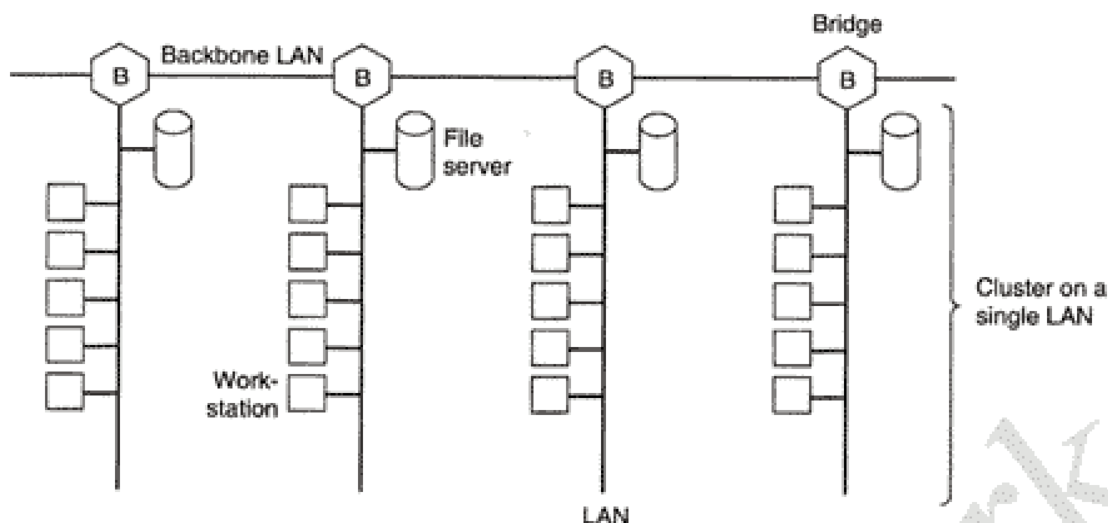
قبل از پرداختن به تکنولوژی پل ، بررسی شرایطی که در آن ، استفاده از پلها سودمند است، خالی از لطف نخواهد بود. شش دلیل ارائه می‌کنیم که چرا یک سازمان واحد، ممکن است دارای چندین LAN باشد.

اوّل آنکه بسیاری از دانشگاه‌ها و بخشهای مختلف شرکتها ، LAN مختص به خود را دارند تا بتوانند کامپیوترهای شخصی ، ایستگاه‌های کاری (Workstation) و سرویس‌دهنده‌های خاص خود را به هم متصل کنند. از آنجایی که بخشهای مختلف یک موسسه، اهداف متفاوتی را دنبال می‌کنند لذا در هر بخش، فارغ از آنکه دیگر بخشها چه می‌کنند، LAN متفاوتی پیاده می‌شود. دیر یا زود نیاز می‌شود که این LANها با یکدیگر تعامل و ارتباط داشته باشند. دراین مثال، پیدایش LANهای متعدد ناشی از اختیار و آزادی مالکان آن بوده است.

دوّم آنکه ممکن است سازمانها به صورت جغرافیائی در ساختمانهائی با فاصله قابل توجه، پراکنده باشند. شاید داشتن چندین LAN مجزا در هر ساختمان و وصل آنها از طریق «پلها» و لینک‌های لیزری ارزانتر از کشیدن یک کابل واحد بین تمام سایتها تمام شود.

سوم آنکه گاهی برای تنظیم بار و تعدیل ترافیک ، لازم است که یک LAN منطقی و واحد به چندین LAN کوچکتر تقسیم شود. به عنوان مثال در بسیاری از دانشگاه‌ها هزاران ایستگاه کاری در اختیار دانشجویان و هیئت علمی قرار گرفته است. عموماً فایلها در ماشینهای سرویس‌دهندهٔ فایل نگهداری می‌شوند و حسب تقاضای کاربران برروی ماشینشان منتقل و بارگذاری می‌شود. مقیاس بسیار بزرگ این سیستم مانع از آن می‌شود که بتوان تمام ایستگاه‌های کاری را در یک شبکهٔ محلی واحد قرار داد چرا که پهنای باند مورد نیاز بسیار بالا خواهد بود. در عوض، مشابه با شکل ۴-۳۹ از چندین LAN که توسط «پل» به هم متصل شده، استفاده می‌شود. هر شبکهٔ LAN گروهی از ایستگاه‌ها و سرویس دهندهٔ فایل خاص خود را دربرمی‌گیرد که بدین ترتیب، بیشتر ترافیک در حوزهٔ یک LAN واحد محدود می‌شود و بار زیادی به ستون فقرات شبکه اضافه نخواهد شد.

اشاره به این نکته ارزشمند است که اگرچه عموماً شبکه‌های LAN را به صورت یک کابل چنداتصالی (Muitidrop) با ساختار باس ترسیم می‌کنیم (نمایش کلاسیک) ولیکن امروزه اغلب آنها توسط هاب و خصوصاً سوئیچها پیاده‌سازی می‌شوند. با این حال یک کابل طولانی با چندین ماشین متصل به آن، با یک هاب که ماشینها را از درون ، بهم متصل می‌کند، از لحاظ عملکرد یکسان هستند. در هر دو حالت تمام ماشینها به یک «حوزهٔ تصادم» (Collision Domain) یکسان متعلقند و تمام آنها برای ارسال فریم از پروتکل CSMA/CD استفاده می‌کنند.



شکل ۴ـ۳۹. چندین LAN از طریق یک ستون‌فقرات بهم متصل شده‌اند تا ظرفیت کلّ حمل بار آن از ظرفیت یک LAN واحد بیشتر شود.

شبکه‌های مبتنی بر سوئیچ متفاوت هستند و اگر چه قبلاً آنها را بررسی کرده‌ایم ولی باز هم نگاهی به آنها خواهیم انداخت .

چهارم آنکه در برخی از شرایط اگرچه یک شبکهٔ محلی واحد از نظر حجم بار کفایت می‌کنند ولیکن فاصلهٔ فیزیکی بین ماشینهای دور، بسیار زیاد است. (مثلاً بیش از ۲/۵ کیلومتر در اترنت). حتی اگر عملیات کابل‌کشی ساده باشد ولیکن شبکه، در اثر تاخیر بسیار زیاد رفت و برگشت سیگنال (Round Trip Delay) کار نخواهد کرد. تنها راه حَل آنست که LAN به چند بخش تقسیم شده و بین آنها پل نصب گردد. با استفاده از پل ، می‌توان فاصلهٔ فیزیکی کل شبکه را افزایش داد.

پنجم مسئله قابلیت‌اعتماد است؛ برروی یک LAN واحد، یک گره خراب که دنباله‌ای پیوسته از خروجی آشغال تولید می‌کند قادر است کّل شبکه را فلج نماید. پلها را می‌توان در نقاط حساس قرار داد تا یک گره خراب و مغشوش نتواند کّل سیستم را مختل کند. برخلاف یک تکرارکننده (Repeater) که ورودی خود را بی قید و شرط بازتولید می‌نماید یک پل را می‌توان به نحوی برنامه‌ریزی کرد تا در خصوص آنچه که هدایت می‌کند یا هدایت نمی‌کند تصمیم آگاهانه بگیرد.

ششم و آخر آنکه پلها می‌توانند به امنیت اطلاعات در یک سازمان کمک نمایند. بیشتر کارتهای واسط شبکه‌های LAN دارای حالتی به نام «حالت بی‌قید» (Promiscuous mode) هستند که در چنین حالتی تمام فریمهای جاری برروی شبکه تحویل گرفته می‌شود، نه فریمهائی که دقیقاً به آدرس او ارسال شده‌اند. جاسوسان و فضولان به این ویژگی علاقمند هستند. با قرار دادن پلها در نقاط مختلف و اطمینان از عدم هدایت اطلاعات حساس به بخش‌های نامطمئن، مسئول سیستم می‌تواند بخشهائی از شبکه را از دیگر بخشها جدا کرده تا ترافیک آنها به خارج راه پیدا نکرده و در اختیار افراد نامطمئن قرار نگیرد.

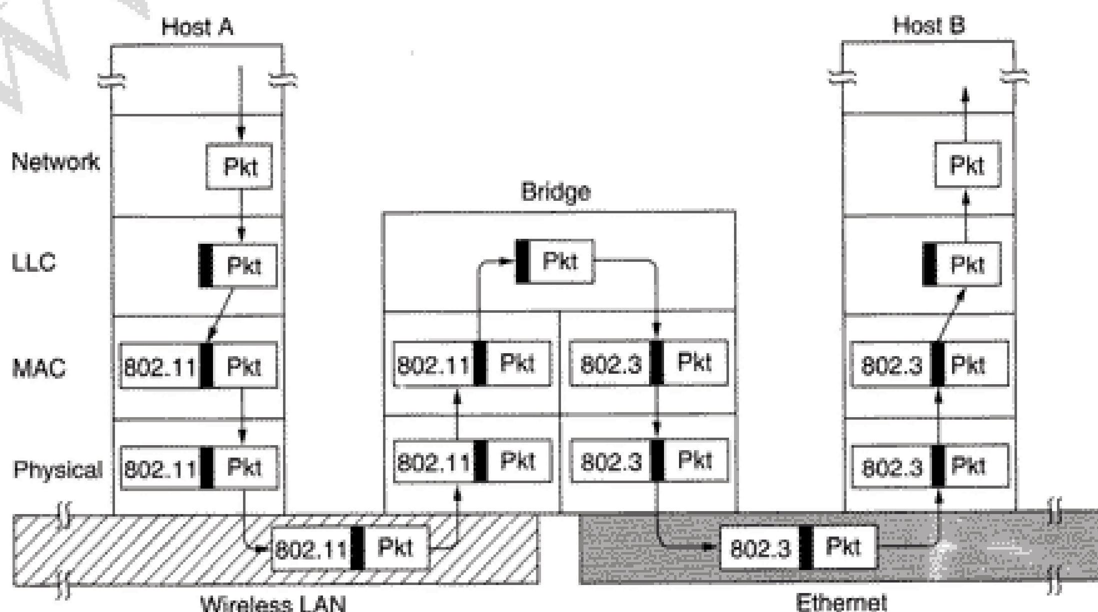
هدف آرمانی آنست که پلها کاملاً نامرئی (شفاف–transparent) باشند، بدین معنا که بتوان ماشینی را از یک بخش از شبکه به بخش دیگر منتقل کرد بدون آنکه به هیچگونه تغییری در سخت‌افزار، نرم‌افزار یا جداول پیکربندی نیاز باشد. همچنین باید این امکان وجود داشته باشد که تمام ماشینهای یک بخش از شبکه بتوانند فارغ از آنکه نوع LAN آنها چیست با ماشینهای بخش دیگر، مبادلهٔ اطلاعات داشته باشند. این هدف گاهی برآورده می‌شود ولیکن نه همیشه!

۴-۷-۱ پلهائی از 802.x به 802.y

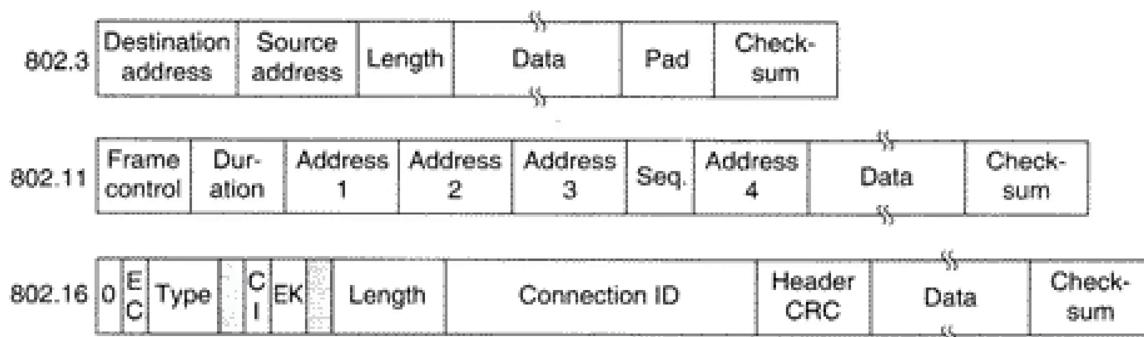
پس از بررسی آنکه چرا به پلها نیاز است اجازه بدهید بدین سوال بپردازیم که عملکرد آنها چگونه است؟ شکل ۴-۴۰ عملکرد یک پل ساده با دو درگاه (Port) را به تصویر کشیده است. ماشین میزبان A در یک شبکهٔ محلی بی‌سیم، بسته‌ای برای ارسال به ماشین میزبان و ثابت B در شبکهٔ اترنت (802.3) (که از طریق پل به شبکهٔ بی‌سیم متصل شده) آماده کرده است. این بسته از زیرلایهٔ LLC عبور کرده و سرآیند LLC به آن اضافه می‌شود (این سرآیند در شکل به صورت سیاه نشان داده شده است). سپس به زیرلایهٔ MAC تحویل شده و در آنجا نیز سرآیند 802.11 به‌ابتدای آن افزوده می‌شود. (یک بخش انتهایی نیز به آخر فریم اضافه خواهد شد که در شکل نشان داده نشده است.) این واحد داده ، در هوا منتشر و توسط ایستگاه ثابت دریافت می‌شود؛ پس از بررسی ، ایستگاه ثابت متوجه می‌گردد که باید آنرا به سمت اترنت ثابت هدایت کند. پس از رسیدن آن به پل (که شبکهٔ 802.11 را به شبکهٔ 802.3 متصل کرده)، پل کار دریافت آن از لایهٔ فیزیکی را شروع کرده و آنرا به سمت زیرلایه‌های بالا هدایت می‌کند. در زیرلایهٔ MAC از پل، سرآیند 802.11 حذف می‌شود. بستهٔ اصلی (به همراه سرآیند LLC) تحویل زیرلایهٔ LLC از پل می‌شود. در این مثال، بسته به سمت شبکه محلی 802.3 رهسپار است، لذا بسته از طریق بخش 802.3 در پل به سمت پائین حرکت کرده و نهایتاً برروی شبکهٔ اترنت منتقل می‌شود. دقت کنید که یک پل که k شبکهٔ مختلف را به هم متصل می‌کند دارای k زیرلایهٔ MAC و تعداد k لایهٔ فیزیکی است (به ازای هر نوع یکی).

تا اینجا به نظر می‌رسد که انتقال فریم از یک LAN به LAN دیگر ساده است اما واقعیت این چنین نیست. در این بخش، برخی از دشواریهای ساخت یک پل را که انواع مختلف شبکه‌های LAN (و همچنین شبکهٔ MAN سری 802 را به هم متصل می‌کند، متذکر می‌شویم. ما برروی شبکه‌های 802.3، 802.11 و 802.16 متمرکز خواهیم شد ولی انواع دیگر آن هم وجود دارد که هر کدام مسائل خاص خود را دارند.

برای شروع باید یادآوری کرد که هر LAN قالب فریم خاص خود را دارد. (شکل ۴-۴۱ را ببینید) اگرچه تفاوتهایی که بین قالب فریم در شبکه‌های اترنت ، «توکن رینگ» (Token Ring) و «توکن باس» (Token Bus) وجود دارد بیشتر ناشی از اتکاء به نفس شرکتهای بزرگ ابداع‌کنندهٔ آنها (یعنی به ترتیب زیراکس، آی‌بی‌ام و جنرال‌موتورز) و همچنین شرایط زمانی آن دوران بوده ولیکن تفاوت شبکه‌های کنونی تقریباً لازم و



شکل ۴-۴۰. عملکرد یک پل از شبکهٔ 802.11 به 802.3.



شکل ۴ـ۴۱. انواع قالب فریمهای IEEE 802 (طول هر فریم در شکل، مقیاس اندازهٔ واقعی آن نیست.)

واقعی است. به عنوان مثال فیلد Duration (طول زمان) که در 802.11 وجود دارد ناشی از ماهیت پروتکل MACAW و همچنین عدم توانایی شنود کانال در این شبکه می‌باشد (برخلاف اترنت). در نتیجه، انتقال یک فریم بین دو شبکهٔ متفاوت LAN، مستلزم دگرگونی در قالب فریمهاست که طبعاً نیاز به زمان CPU، محاسبهٔ مجدّد کدهای جدید کشف خطا (Checksum) دارد و این احتمال نیز پدید می‌آید که در اثر خرابی بیتها در حافظه پل، خطاهای غیرقابل کشف، داده‌ها را آلوده کند.

مشکل دوم آنست که LANهای به هم متصل‌شده الزاماً با سرعت مشابهی کار نمی‌کنند. وقتی «پل» یک دنبالهٔ پی‌درپی از فریمها را از LAN سریع برای هدایت به LAN کندتر می‌پذیرد قادر نخواهد بود با همان سرعتی که فریم را دریافت می‌کند از دست آنها رهائی یابد. به عنوان مثال هرگاه یک شبکهٔ اترنت گیگابیتی با بالاترین سرعت، بیتها را به سوی شبکهٔ یازده مگابیت‌درثانیه‌ای 802.11b LAN روانه کند، پل باید بتواند آنها را موقتاً در حافظه نگهداری کند و این حافظه نباید سرریز شود. [که با چنین سرعتی بعید نیست.] در ضمن، برخی پلها سه یا چندین شبکهٔ LAN را به هم متصل می‌کنند. در چنین پلهایی اگر چندین شبکه بطور همزمان تلاش کنند فریمهای خود را به یک LAN مشابه بفرستند مشکل سرریز شدن حافظه بوجود خواهد آمد، حتی اگر تمام LANها با سرعت یکسانی کار کنند.

مشکل سوّم که خطیرترین مشکل بالقوهٔ پلها محسوب می‌شود آنست که در شبکه‌های محلی و مختلف سری 802، حداکثر طول فریم، متفاوت است. این مشکل زمانی بروز می‌کند که یک فریم بزرگ باید به سوی شبکه‌ای هدایت شود که آن LAN قادر به دریافت آن نیست. شکستن فریم به تعدادی قطعه، خارج از حیطهٔ وظائف این لایه است. در تمام پروتکلهای این لایه فرض برآنست که فریمها یا می‌رسند یا نمی‌رسند و هیچ تمهیدی برای شکستن یک فریم به قطعات و بازسازی آن اندیشیده نشده است. نمی‌توان گفت که چنین پروتکلی ابداع نشده است. چنین پروتکلی ابداع شده و وجود هم دارد ولیکن در حیطه وظایف پروتکلهای پیونده داده نیست. پلها نیز نباید به محتوای هر فریم کاری داشته باشند. اساساً (در چنین وضیعتی) راه حلی برای این مشکل وجود ندارد. فریمهای بسیار طولانی باید به جای انتقال، حذف شوند. [جهت اجتناب از هرگونه مداخله در پیکربندی سخت‌افزار یا نرم‌افزار]

نکته بعدی امنیت داده‌هاست. شبکه‌های 802.11 و 802.16، هر دو از رمزنگاری در سطح لایهٔ پیوند داده‌ها پشتیبانی می‌کنند در حالیکه اترنت چنین امکانی ندارد. بدین معنا که خدمات متنوع رمزنگاری عرضه شده در شبکه‌های بی‌سیم، در خلال ورود ترافیک به شبکهٔ اترنت از دست می‌رود. بدتر از آن، اینکه اگر یک دستگاه در شبکهٔ بی‌سیم از رمزنگاری در سطح لایهٔ پیوند داده‌ها بهره گرفته باشد هیچ راهی برای رمزگشائی آن هنگام

دریافت در یک ایستگاه اترنت وجود ندارد. از طرفی اگر ایستگاه بی‌سیم از رمزنگاری استفاده نکند ترافیک داده‌ها از طریق لینک هوائی در معرض شنود همگان قرار می‌گیرد.

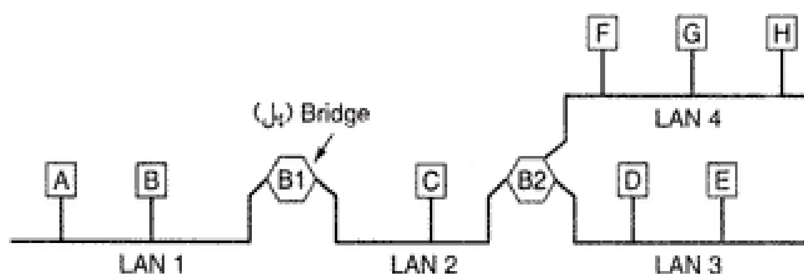
یک راه‌حل برای مشکل امنیت آنست که رمزنگاری در لایه‌های بالائی انجام شود ولیکن در این روش ایستگاه 802.11 باید بداند که آیا با ایستگاه دیگری در شبکهٔ 802.11 صحبت می‌کند (تا از رمزنگاری در لایهٔ پیوند داده بهره بگیرد) یا نه (تا از چنین امکانی استفاده نکند). وادار کردن ایستگاه به تصمیم‌گیری «شفافیت» را از بین خواهد برد.

نکتهٔ آخر مسئله «کیفیت خدمات» است. هر دو شبکهٔ 802.11 و 802.16 این خدمات را به نحو متفاوتی در اختیار می‌گذارند: اوّلی در «حالت PCF» و دوّمی با استفاده از «اتصال با نرخ ارسال ثابت» (Constant Bit Rate Connection). در شبکهٔ اترنت چیزی به نام کیفیت خدمات بی‌مفهوم است و بدین ترتیب ترافیک هر یک از این دو شبکه در حین عبور از اترنت «کیفیت خدمات» را از دست می‌دهند. [به عنوان مثال اگر چه می‌توان میزان حداکثر تاخیر در دو شبکهٔ اوّل را تضمین کرد ولی در اترنت چنین تضمینی وجود ندارد و با وصل این دو شبکه نمی‌توان تاخیر کلّ را تضمین نمود و مقدار آن تصادفی خواهد بود. -م]

۴-۷-۲ بهم‌بندی شبکه‌ها به صورت محلی (Local Internetworking)

در بخش قبلی به مشکلاتی که در وصل دو شبکهٔ محلّی نوع IEEE 802 بروز می‌کند، پرداختیم. با این وجود در سازمانهای بزرگ با شبکه‌های LAN متعدد، متصل کردن تمام آنها به یکدیگر مشکلات گوناگونی را بوجود می‌آورد، حتی اگر تمام آنها از نوع اترنت باشند. حالت آرمانی آنست که بتوان به راحتی از سازمان بیرون رفت و چند پل مبتنی بر استاندارد IEEE خریداری کرد، سپس با وصل تمام کابلها به پلها ، شبکه فوراً و به درستی بکار بیفتد. نباید به هیچ تغییر سخت‌افزاری، نرم‌افزاری، تنظیم سوئیچها، بارگذاری یا تغییر در جداول مسیریابی یا پارامترهای آن یا هر تغییر دیگری نیاز باشد. فقط باید کابلها را وصل کرد و رفت!! به علاوه عملکرد طبیعی هیچکدام از شبکه‌های LAN نباید تحت‌الشعاع قرار بگیرد. به عبارت دیگر، پلها بایستی کاملاً «شفاف» باشند (یعنی از دیدگاه سخت‌افزار و نرم‌افزار غیرقابل رویت باشند). شگفت آنکه این کار عملاً ممکن است. حال اجمالاً بررسی کنیم که این کار جادوئی به چه نحو انجام می‌شود!

در ساده‌ترین حالت، یک «پل شفاف» در «حالت بی‌قید» (Promiscuous Mode) عمل کرده و تمام فریمهای جاری برروی تمام LANهائی را که بدانها متصل است، می‌پذیرد. به عنوان یک مثال، به پیکربندی شکل ۴-۴۲ توجه نمائید. پل B1 به شبکه‌های محلی ۱ و ۲ متصل شده و پل B2 نیز به شبکه‌های محلی ۲ و ۳ و ۴ متصل است. فریمی که به پل B1 می‌رسد ولی مقصد آن ماشین A است می‌تواند فوراً توسط پل نادیده انگاشته شود. (حذف شود) زیرا این فریم برروی LAN صحیحی [که به مقصد ختم می‌شود] قرار گرفته است ولی فریمی که از LAN 1 به مقصد C یا F دریافت می‌شود باید منتقل شود.



شکل ۴-۴۲. یک پیکربندی از شبکه‌های متصل‌بهم با چهار شبکهٔ محلی و دو پل.

وقتی فریمی دریافت می‌شود، پل در ابتدا باید تصمیم بگیرد که آیا باید آنرا حذف کند یا باید آنرا منتقل نماید؛ سپس در صورت نیاز به انتقال، باید مشخص شود که به کدام LAN هدایت گردد. این تصمیم‌گیری با جستجوی آدرس مقصد درون یک جدول بزرگ در حافظهٔ پل انجام می‌گیرد. (به این جدول ، Hash Table گفته می‌شود.) این جدول فهرست تمام ماشینهای مقصد را در اختیار دارد و می‌تواند تعیین کند که این ماشینها به کدامیک از خطوط پل تعلق دارند. مثلاً جدول پل B2 می‌تواند مشخص کند که ماشین A به LAN 2 تعلق دارد، زیرا تمام آنچه که لازم است B2 بداند آنست که فریمهائی با مقصد A را برروی چه شبکه‌ای ارسال کند. در واقع برای پل B2 مهم نیست که این فریم، بعداً چگونه هدایت و منتقل می‌شود.

وقتی پلها برای اولین بار به کار می‌افتند تمام جداول Hash خالی هستند. هیچیک از پلها نمی‌دانند که هر یک از ماشینهای مقصد در کجا قرار گرفته‌اند، لذا برای انتقال فریم از **«الگوریتم سیل‌آسا»** (Flooding Algorithm) استفاده می‌نماید یعنی تمام فریمهای ورودی که مقصدشان ناشناخته است برروی تمام شبکه‌هائی که پل بدانها متصل است، ارسال می‌شود (البته به استثنای شبکه‌ای که فریم از آن دریافت شده است). به مرور زمان، پل متوجه خواهد شد که هر ماشین مقصد، در کجا قرار گرفته است. (طبق الگوریتمی که در زیر بدان اشاره خواهیم کرد.) هرگاه یک ماشین مقصد شناسائی شد، فریمهائی که بعداً بدان مقصد روانه می‌شوند توسط پل برروی LAN مناسب هدایت خواهد شد و از روش سیل‌آسا استفاده نمی‌شود.

الگوریتمی که توسط پلهای شفاف به کار گرفته می‌شود روش **«یادگیری غیرمستقیم»** (Backward Learning) است. قبلاً اشاره شد که پل در «حالت بی‌قید» کار می‌کند لذا هر فریمی را که برروی یکی از شبکه‌های متصل به او مبادله می‌شود، می‌بیند (و همچنین دریافت و پردازش می‌نماید). با نگاهی به آدرس مبداء هر فریم، پل می‌تواند بفهمد که کدام ماشین از طریق کدام LAN قابل دسترسی است. به عنوان مثال در شکل ۴-۴۲ فرض کنید که پل B1 فریمی را بر روی LAN 2 می‌بیند که توسط C تولید شده است؛ بدین ترتیب متوجه می‌شود که می‌توان از طریق LAN 2 به C رسید و بدین ترتیب در جدول Hash خود درج می‌کند که فریمهای روانه به سمت C باید از طریق LAN 2 منتقل و هدایت شوند. بعداً تمام فریمهائی که به مقصد C و از طریق LAN 1 به پل وارد می‌شوند منتقل می‌شوند ولی فریمهائی که به مقصد C ولی از طریق LAN 2 به پل می‌رسند حذف خواهند شد.

با خاموش یا روشن شدن پلها و ماشینها و یا جابجائی آنها در سطح شبکه، توپولوژی شبکه تغییر می‌کند. برای آنکه توپولوژی شبکه به صورت پویا دنبال شود وقتی یک **«درایه»** (Entry) در جدول هر پل درج می‌گردد زمان دریافت فریم نیز در آن «درایه» یادداشت می‌شود. هرگاه بعداً فریمی که آدرس مبداء آن قبلاً در جدول درج گردیده، دریافت شود زمان درج شده در درایهٔ متناظر آن، با زمان فعلی بهنگام‌سازی می‌شود. بدین ترتیب زمان درج شده در هر «درایه» (Entry) آخرین زمانی که فریمی از آن ماشین دریافت شده را تعیین می‌کند.

بطور متناوب، یک پروسه در درون پل، جدول Hash را جستجو و پویش می‌کند و تمام درایه‌هائی را که برای بیش از چند دقیقه به هنگام نشده‌اند، حذف می‌نماید. بدین ترتیب اگر یک کامپیوتر از شبکهٔ خود جدا و در ساختمان جابجا شده یا به شبکهٔ دیگری متصل شود پس از گذشت چند دقیقه عملیات عادّی خود را از سرمی‌گیرد و نیازی به مداخله و تنظیمات دستی نیست. البته این الگوریتم بدین معنا هم هست که اگر کامپیوتری برای چندین دقیقه ساکت باشد (یعنی هیچ داده‌ای نفرستد) درایهٔ متناظر با او از جدول پلها حذف شده و از آن به بعد هر ترافیکی که برایش ارسال می‌شود بروش سیل‌آسا هدایت خواهد شد مگر آنکه بعداً خودش فریمی ارسال کند [و آدرس او در جدول پلها درج شود. -م]

روند مسیریابی فریمهای ورودی پل ، به شبکهٔ LAN مبداء (Source LAN) و شبکهٔ LAN مقصد (Destination LAN) بستگی دارد. این روال به ترتیب ذیل است:

۱. اگر شبکهٔ LAN مبدأ و مقصد یکسان هستند فریم را نادیده بگیر.

۲. اگر شبکه‌های مبدأ و مقصد متفاوت هستند فریم را منتقل کن.

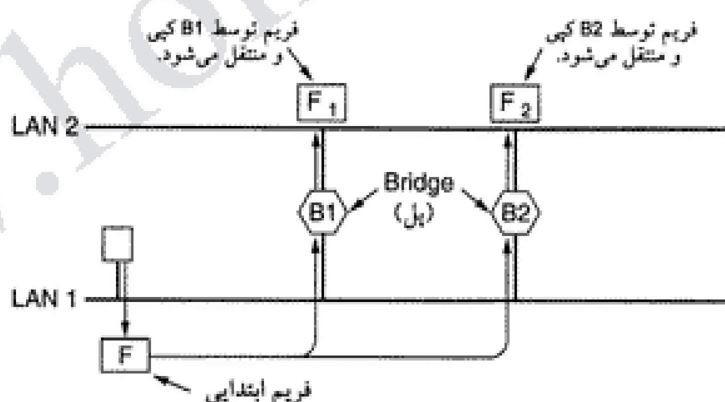
۳. اگر شبکهٔ مقصد ناشناخته است بروش «سیل‌آسا» عمل کن.

این الگوریتم به ازای ورود هر فریم باید یکبار اعمال گردد. یک تراشهٔ خاص VLSI عملیات جستجو و به هنگام‌سازی جدول و درایه‌های (Entries) را در عرض چند میکروثانیه انجام می‌دهد.

۴-۷-۳ پلهای مبتنی بر درخت پوشا (Spanning Tree)

برخی از سایتها برای افزایش قابلیت اعتماد، بین دو LAN دو یا چند پل موازی نصب می‌کنند.(به شکل ۴-۴۳ نگاه کنید.) ولیکن چنین آرایشی مشکلات دیگری را ایجاد خواهد کرد چراکه در ساختار توپولوژی، حلقه ایجاد می‌شود.

چنین مشکلاتی را می‌توان بکمک مثال شکل ۴-۴۳ و با مشاهدهٔ روند هدایت فریمی از ماشین F به مقصدی ناشناخته، تحلیل کرد. هر پل طبق قاعدهٔ عمومی در مواجهه با فریمهائی که به مقصد ناشناخته روانه هستند از روش «سیل‌آسا» استفاده می‌کند یعنی در اینجا فریم برروی LAN 2 منتقل می‌شود. [فریم منتقل شده بر روی LAN 2 را F_2 بنامید.] اندکی بعد، پل ۱ فریم F_2 را می‌بیند که مقصد آن ناشناخته است و آنرا برروی LAN 1 هدایت می‌کند و فریم F_3 تولید می‌شود. (فریم F_3 در شکل نشان داده نشده است.) بروش مشابه پل ۲ فریم F_1 را برروی LAN 1 منتقل کرده و F_4 تولید می‌شود. (F_4 نیز نشان داده نشده است) [F_1، F_2، F_3، F_4 نسخه‌های مشابه با فریم F هستند که توسط پل منتقل می‌شوند.] حال پل ۲ فریم F_4 را منتقل و پل ۲ فریم F_3 را منتقل می‌نماید و این دور باطل تا ابد ادامه دارد.



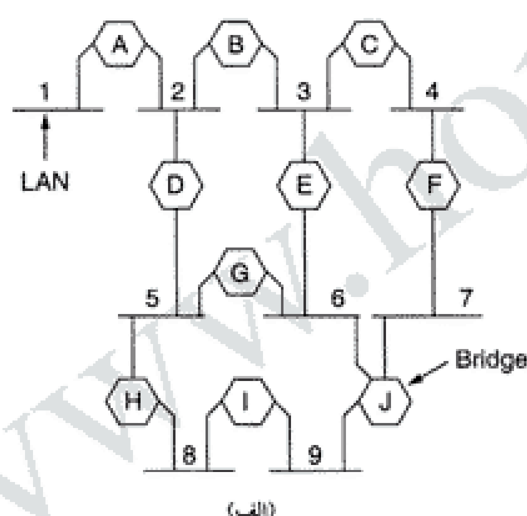
شکل ۴-۴۳. دو پل شفاف (نامرئی) موازی.

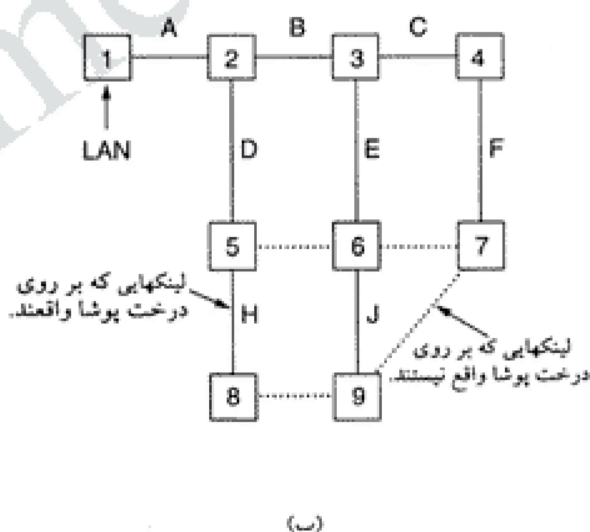
راه حلّ این مشکل آنست که پلها با یکدیگر ارتباط و محاوره داشته باشند و توپولوژی واقعی را به صورت یک «**درخت پوشا**» (Spanning Tree) که در آن به تمام LANها مسیر دسترسی وجود دارد، در نظر بگیرند. در نتیجه برخی از اتصالات بالقوه که بین LANها وجود دارد، نادیده گرفته می‌شود تا مجازاً یک توپولوژی بدون حلقه ایجاد شود. به عنوان مثال در شکل ۴-۴۴-الف ، نُه شبکهٔ LAN را می‌بینیم که توسط ده پل متصل شده‌اند. این پیکربندی را می‌توان در قالب گرافی در نظر گرفت که در آن، هر کدام از LANها یک «گره» تلقی می‌شوند. هر «کمان» (Arc) اتصال دو LAN توسط پل را نشان می‌دهد. با حذف برخی از «کمانها» (که در شکل ۴-۴۴-ب به صورت خط چین نشان داده شده) ، این گراف به یک درخت پوشا کاهش می‌یابد. با استفاده از این درخت، بین دو LAN دقیقاً یک مسیر وجود دارد. پس از آنکه پلها درخت پوشا را تشکیل دادند، هدایت تمام فریمها (بین

شبکه‌های محلّی) از ساختار درخت پوشا تبعیت خواهد کرد. از آنجایی که بین هر مبداء و مقصد در شبکه فقط و فقط یک مسیر وجود دارد لذا ایجاد حلقه غیرممکن است.

برای ایجاد درخت پوشا، پلها بایستی یک پل را به عنوان ریشهٔ این درخت انتخاب کنند. جهت این انتخاب، پلها شمارهٔ سریال خود را که توسط کارخانهٔ سازنده تنظیم و یکتا بودن آن در کلّ دنیا تضمین شده است، به صورت فراگیر (Broadcast) به اطلاع همه می‌رسانند. پلی که دارای کوچکترین شمارهٔ سریال است به عنوان «ریشه» انتخاب می‌شود. سپس درختی با کوتاهترین مسیر که از ریشه شروع شده و تمام پلها و LANها را در بر می‌گیرد، ایجاد می‌شود. این درخت همان «درخت پوشا» است. اگر یک پل یا شبکهٔ LAN از کار بیفتد، این درخت از نو محاسبه می‌شود.

نتیجهٔ این الگوریتم آنست که بین هر LAN و ریشه و بدین ترتیب از ریشه به بقیهٔ LANها، یک مسیر یکتا ایجاد می‌شود. اگر چه این درخت تمام LANها را در بر می‌گیرد ولیکن تمام پلها لزوماً در این درخت قرار نمی‌گیرند (برای اجتناب از حلقه). حتی پس از ایجاد درخت پوشا و در خلال عملکرد طبیعی، این الگوریتم به صورت متناوب اجرا می‌شود تا هر گونه تغییر در توپولوژی به صورت خودکار تشخیص داده شده و درخت اصلاح شود. این الگوریتم توزیع‌شده که برای ایجاد درختهای پوشا مورد استفاده قرار می‌گیرد توسط خانم رایدیا پرلمن ابداع و به تفصیل در مرجع (Perlman,2000) تشریح شده است. همچنین این الگوریتم در IEEE 802.1D استانداردسازی شده است.



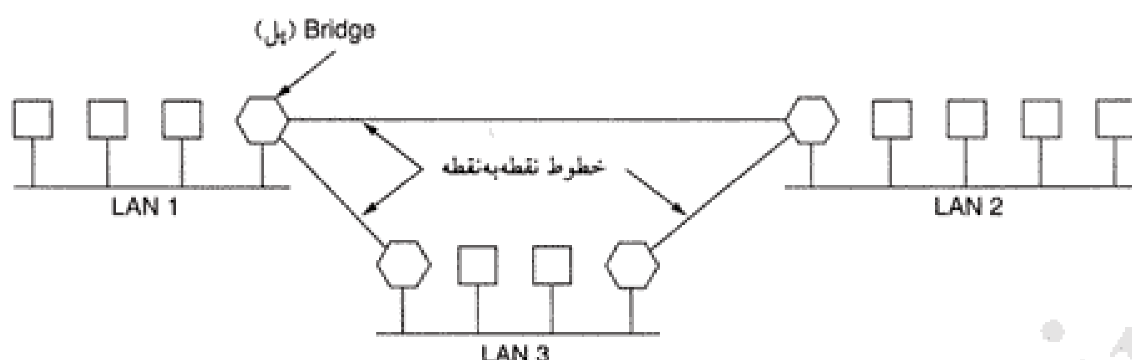


شکل ۴ـ۴۴. (الف) چند شبکهٔ LAN بهم متصل (ب) یک درخت پوشا که تمام شبکه‌های LAN را در بر می‌گیرد. (خطوط نقطه‌چین جزو درخت پوشا نیستند.)

٤ـ٧ـ٤ پلهای راه دور (Remote Bridges)

کاربرد متداول پلها آنست که دو یا چند LAN راه دور را بهم متصل کنند. بعنوان مثال ممکن است یک شرکت دارای کارخانه‌هایی در چند شهر باشد و هر کارخانه، LAN مختص به خود را داشته باشد. در حالت آرمانی باید تمام LANها به هم متصل شده باشند تا کلّ سیستم نقش یک LAN عظیم را ایفاء کند.

این هدف با قرار دادن یک پل بین هر دو LAN و وصل کردن پلها به وسیلهٔ خطوط نقطه‌به‌نقطه (مثلاً بکمک خطوط اجاره‌ای عرضه شده توسط شرکتهای تلفن) برآورده خواهد شد. در شکل ۴-۴۵، یک سیستم ساده با سه LAN به تصویر کشیده شده است. در اینجا روشهای رایج مسیریابی اعمال می‌شود. ساده‌ترین راه برای تحلیل این



شکل ۴ـ۴۵. برای اتصال شبکه‌های محلی راه دور می‌توان از پلهای راه دور بهره گرفت.

ساختار آنست که سه خط نقطه‌به‌نقطه را به مثابه یک شبکهٔ LAN بدون هیچ ماشین میزبان (Hostless LAN) ، تصور کنید. در این صورت یک سیستم معمولی با شش LAN داریم که با چهار پل به هم متصل شده‌اند. در هیچ کجا از مطالبی که تاکنون مطالعه کرده‌ایم گفته نشده که حتماً باید به یک LAN ماشین میزبان متصل شده باشد!

برای خطوط نقطه‌به‌نقطه می‌توان از پروتکلهای متنوعی بهره گرفت. یک انتخاب آنست که از یکی از استانداردهای موجود برای خطوط نقطه‌به‌نقطه مثل PPP استفاده شود و کلّ فریمهای MAC [شامل سرآیند و فیلدهای پایانی] درون فیلد حمل دادهٔ آن (یعنی Payload) قرار بگیرد. این استراتژی بشرط آنکه تمام LANها مثل هم باشند، به نحو احسن کار می‌کند و تنها مسئلهٔ باقیمانده آنست که فریمها به LAN صحیح تحویل شوند. انتخاب دیگر آنست که سرآیند و پی‌آیند[1] هر فریم MAC در پل مبداء حذف شود و باقیمانده در فیلد حمل داده از فریم مربوط به پروتکل نقطه‌به‌نقطه قرار بگیرد. سپس در پل مقصد سرآیند و پی‌آیند MAC جدیدی برای آن قطعهٔ داده تولید شود. اشکال این روش آنست که کد کشف خطایی که فریم در حین دریافت در ماشین مقصد دارد همانی نیست که توسط ماشین مبداء تولید شده و بدین ترتیب ممکن است خطاهایی که در حافظهٔ پل ، داده‌ها را آلوده می‌کند کشف نشود.

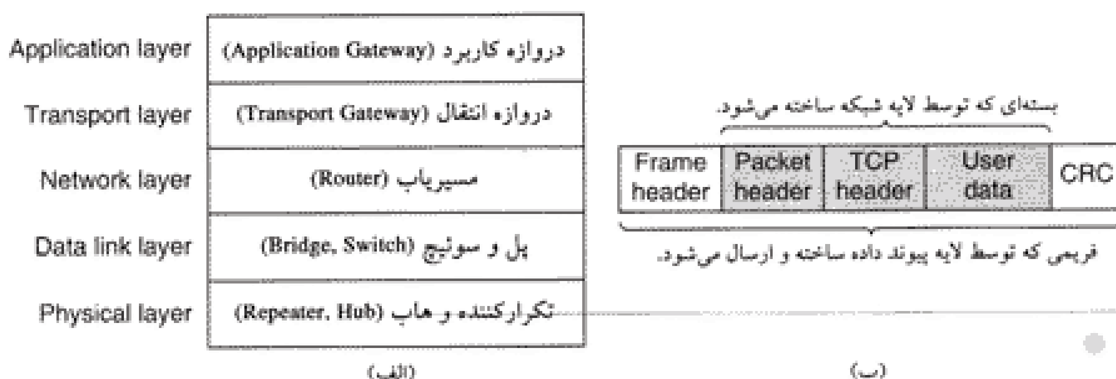
۴ـ۷ـ۵ تکرارکننده[2]، هاب[3]، پل[4]، سوئیچ[5]، مسیریاب[6] و دروازه[7]

تا اینجای کتاب روشهای گوناگونی را مرور کردیم که وظیفهٔ همگی تحویل فریمها یا بسته‌ها از یک بخش کابل به بخش دیگر است. همچنین اشاره‌ای به تکرارکننده‌ها، پلها، سوئیچها، هابها، مسیریابها و دروازه‌ها داشتیم. از تمام این ابزارها به یک منظور استفاده می‌شود ولیکن تفاوتهای مشهود و نامشهود زیادی دارند. از آنجایی که این ابزارها بسیار متنوعند، لذا مروری بر همهٔ آنها و بررسی شباهتها و تفاوتهای آنها ارزشمند خواهد بود.

برای شروع باید گفت که این ابزارها (به نحوی که در شکل ۴-۴۶-الف دیده می‌شود) در لایه‌های متفاوتی عمل می‌کنند. بسته به آنکه هر یک از این ابزارها در چه لایه‌ای عمل می‌کنند مکانیزم هدایت اطلاعات متفاوت است. در قالب یک نمایشنامهٔ عمومی: کاربر مقداری اطلاعات برای ارسال به یک ماشین راه دور تولید می‌نماید. این داده‌ها تحویل لایهٔ انتقال (Transport Layer) شده و بدان سرآیند لازم اضافه می‌گردد (مثلاً سرآیند TCP). سپس، واحد اطلاعاتی حاصل به سمت لایهٔ پائین یعنی لایهٔ شبکه عبور داده می‌شود. لایهٔ شبکه سرآیند خاصّ خود را بدان افزوده و یک بستهٔ مخصوص لایهٔ شبکه ساخته می‌شود (مثلاً یک بستهٔ IP). در شکل ۴-۴۶-ب یک بستهٔ IP را می‌بینیم که به صورت خاکستری نشان داده شده است. این بسته تحویل لایهٔ پیوند داده (Data Link)

---

۱. Header and Trailer ۲. Repeater ۳. Hub ۴. Bridge
۵. Switch ۶. Router ۷. Gateway

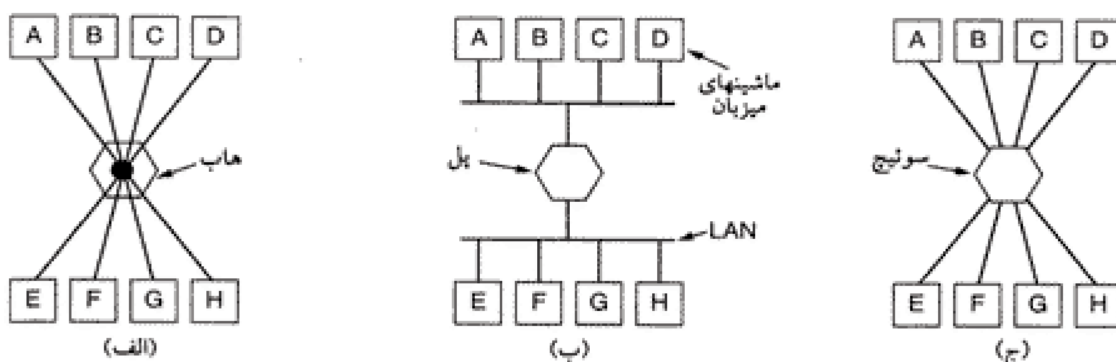


شکل ۴ـ۴۶. (الف) جایگاه هر ابزار در پشتهٔ پروتکلی (ب) فریمها، بسته‌ها و سرآیندها.

می‌شود و آن هم سرآیند خاص خود و کد کشف خطا (CRC) را بدان افزوده و فریم حاصل را جهت ارسال به لایهٔ فیزیکی تسلیم می‌کند. (مثلاً برای ارسال برروی LAN)

حال اجازه بدهید نگاهی به ابزارهای هدایت اطلاعات بیندازیم و ببینیم که این ابزارها چه ارتباطی با بسته‌ها و فریمها دارند. در پائین‌ترین سطح یعنی در لایهٔ فیزیکی به «تکرارکننده‌ها» بر می‌خوریم. تکرارکننده ابزاری است آنالوگ، که دو قطعه کابل را بهم متصل می‌کند. سیگنالی که برروی یکی از این قطعات ظاهر گردد، تقویت (بازتولید) شده و بر روی قطعه دیگر قرار داده می‌شود. تکرارکننده‌ها هیچگونه درکی از «فریم»، «بسته» یا «سرآیند» ندارند. آنها صرفاً با مفهوم «وُلت» آشنا هستند! به عنوان مثال در اترنت کلاسیک اجازه داده شده برای افزایش طول حداکثر کابل از ۵۰۰ متر به ۲۵۰۰ متر از چهار تکرارکننده استفاده شود.

سپس به هاب (Hub) می‌رسیم: یک هاب معمولی (از نوع غیرفعال)، دارای تعدادی خط ورودی است که این خطوط از لحاظ الکتریکی در درون هاب به هم متصل شده‌اند. فریمی که از یک خط ورودی دریافت می‌شود بر روی خطوط دیگر ارسال خواهد شد. هرگاه دو فریم بطور همزمان به هاب ارسال شوند، تصادم رخ خواهد داد؛ دقیقاً همانند اتفاقی که برروی کابل کواکسیال می‌افتد. بعبارت دیگر، کلّ هاب یک «حوزهٔ تصادم» (Collision Domain) واحد را تشکیل خواهد داد. تمام خطوط ورودی هاب، باید با سرعت یکسانی کار کنند. هابها متفاوت از تکرارکننده‌ها هستند، از آن جهت که هاب (معمولاً) سیگنالهای ورودی را تقویت نمی‌کند و طراحی آنها به گونه‌ای است که چندین کارت واسط خط (Line Card) دارند و هر یک از کارتها خود چندین ورودی دارند ولیکن در مجموع این تفاوتها ناچیز است. همانند تکرارکننده‌ها، هابها نیز آدرسهای 802 [آدرسهای MAC] را بررسی نکرده و به هیچ وجه از آنها استفاده نمی‌کنند. در شکل ۴-۴۷-الف تصویری نمادین از یک هاب نشان داده شده است.



شکل ۴ـ۴۷. (الف) یک هاب (ب) یک پل (ج) یک سوئیچ.

حال اجازه بدهید به سمت لایهٔ پیوند داده یعنی لایه‌ای که در آن «پلها» و «سوئیچها» تعریف شده‌اند حرکت کنیم. قبلاً «پلها» را تا حدّی مطالعه کردیم. یک پل دو یا چند شبکهٔ LAN را همانند شکل ۴-۴۷-ب به هم متصل می‌کند. وقتی فریمی دریافت می‌شود نرم‌افزار درون پل، آدرس مقصد را از سرآیند فریم استخراج و آنرا درون جدول خود جستجو می‌کند تا محلّی را که فریم باید بدانجا ارسال شود، بیابد. در اترنت، این آدرس همان فیلد ۴۸ بیتی آدرس در شکل ۴-۱۷ است. شبیه به هاب، پلهای پیشرفته نیز دارای «کارت‌های خط» (Line Card) هستند که معمولاً چهار تا هشت خط ورودی از یک نوع شبکهٔ معین در آنها تعبیه شده است. یک «**کارت خط اترنت**» (Ethernet Line Card) نمی‌تواند مثلاً فریمهای شبکهٔ توکن رینگ را بپذیرد چرا که نمی‌داند در کجای سرآیند فریم آدرس مقصد را پیدا کند! با این وجود یک «پل» می‌تواند برای انواع شبکه‌های مختلف با سرعت متفاوت، «کارتهای خط» مجزا داشته باشد. در یک «پل» برخلاف هاب، هر خط «حوزهٔ تصادم» خاص خود را دارد. «سوئیچها» شبیه به پلها هستند چرا که هر دوی آنها براساس آدرسهای درون فریم، آنها را مسیریابی و هدایت می‌کنند. در واقع بسیاری از افراد این دو واژه را به صورت معادل به کار می‌برند. تفاوت اصلی آنها در این است که یک سوئیچ شبیه به شکل ۴-۴۷-ج، برای وصل کردن کامپیوترهای منفرد به یکدیگر، کاربرد دارد. طبعاً وقتی ماشین میزبان A در شکل ۴-۴۷-ب می‌خواهد فریمی را برای ماشین B بفرستد، پل اگر چه فریم را دریافت می‌کند ولی آن را نادیده می‌گیرد. برخلاف آن در شکل ۴-۴۷-ج، سوئیچ باید بلافاصله فریم را از A به سمت B هدایت نماید چرا که هیچ راهی برای تحویل فریم به B [جز از طریق سوئیچ] نیست. از آنجایی که معمولاً هر یک از پورتهای یک سوئیچ به یک کامپیوتر منفرد وصل می‌شود لذا یک سوئیچ باید پورتهای بسیار بیشتری (در مقایسه با پلهائی که برای وصل تعداد کمی LAN طراحی شده‌اند) داشته باشد. در ضمن هر یک از «کارت‌های خط» بایستی فضای بافر کافی برای ذخیرهٔ فریمهای دریافتی از هر یک از پورتها در اختیار داشته باشند. از آنجائی هر یک از پورتها «حوزهٔ تصادم» مجزا و متعلق به خود را دارند لذا هیچ فریمی در اثر تصادم از بین نخواهد رفت. با این وجود اگر فریمها با نرخی بیش از ظرفیت سوئیچ، وارد گردند ممکن است فضای بافر پر شده و سوئیچ مجبور به حذف آنها شود.

برای آنکه این مشکل کمی کاهش یابد در سوئیچهای مدرن به محض آنکه فیلد آدرس مقصد از فریم وارد گردید، عمل هدایت و انتقال فریم شروع می‌شود، حتی قبل از آنکه مابقی فریم بطور کامل دریافت شده باشد. البته در صورتی که خط خروجی مورد نظر آزاد باشد. این سوئیچها روش «ذخیره و هدایت» (Store & Forward) را به کارنمی‌برند. [در روش «ذخیره و هدایت» ابتدا کل فریم دریافتی ذخیره شده و سپس عملیات هدایت آغاز می‌شود]. این نوع از سوئیچها که اغلب به نام Cut-Through Switches (سوئیچهای میانبر) مشهورند، کاملاً به صورت سخت‌افزاری پیاده سازی می‌شوند در حالیکه پلها عموماً دارای یک CPU واقعی بوده و عمل «ذخیره و هدایت» را توسط نرم‌افزار انجام می‌دهند. بهر حال چون تمام پلها و سوئیچهای مدرن دارای تراشهٔ مدار مجتمع ویژه‌ای جهت هدایت و انتقال فریمها هستند لذا امروزه تفاوت‌های بین پل و سوئیچ بیشتر به مسائل بازاری بستگی دارد تا مسائل فنی.

تا اینجا تکرارکننده‌ها و هابها را بررسی کرده‌ایم که کاملاً شبیه به هم هستند و همنطور به پلها و سوئیچها پرداختیم که آنها نیز شباهت زیادی به یکدیگر دارند. حال به سوی «مسیریاب» (Router) حرکت خواهیم کرد که با تمام ابزارهای فوق تفاوت دارد. وقتی بسته‌ای به یک مسیریاب وارد می‌شود ابتدا سرآیند و فیلدهای انتهائی فریم حذف شده و سپس بستهٔ جاسازی شده در درون فیلد حمل دادهٔ فریم (Payload) (که در شکل ۴-۴۶ به صورت خاکستری نشان داده شده) ، تحویل نرم‌افزار مسیریابی می‌شود. این نرم‌افزار برای انتخاب خط خروجی از مشخصات واقع در سرآیند بسته استفاده می‌کند. در بسته‌های IP ، سرآیند هر بسته شامل یک آدرس ۳۲ بیتی

(در IPv4) یا ۱۲۸ بیتی (در IPv6) است ولی آدرس ۴۸ بیتی 802 ندارد. نرم‌افزار مسیریابی نمی‌تواند آدرس فریمها را ببیند و حتی نمی‌تواند متوجه شود که بسته از طریق یک LAN دریافت شده یا از روی یک خط نقطه به نقطه. در فصل پنجم مسیریابها و فرآیند مسیریابی را تشریح خواهیم کرد.

در لایهٔ بالاتر به «دروازه‌های انتقال» (Transport Gateway) بر می‌خوریم. این دروازه‌ها ارتباط دو کامپیوتر را که از پروتکلهای اتصال‌گرای متفاوتی در لایهٔ انتقال استفاده می‌کنند، برقرار می‌نماید. به عنوان مثال فرض کنید کامپیوتری که از پروتکل اتصال‌گرای TCP/IP استفاده کرده، می‌خواهد با کامپیوتری که از پروتکل اتصال‌گرای ATM استفاده می‌کند، محاوره نماید. یک دروازهٔ انتقال می‌تواند بسته‌هائی که از طریق یک «اتصال» دریافت می‌شوند را پس از تغییرات لازم بر روی «اتصال» دیگر بفرستد.

در نهایت، «دروازه‌های کاربرد» (Application Gateway) قالب و محتوای داده‌ها را تشخیص می‌دهند و یک «پیام» را به پیامی دیگر ترجمه می‌کنند. بعنوان مثال یک «دروازه پست الکترونیکی» می‌تواند یک پیام اینترنتی [نامهٔ الکترونیکی] را به پیام SMS برای گوشی‌های تلفن همراه ترجمه نماید.

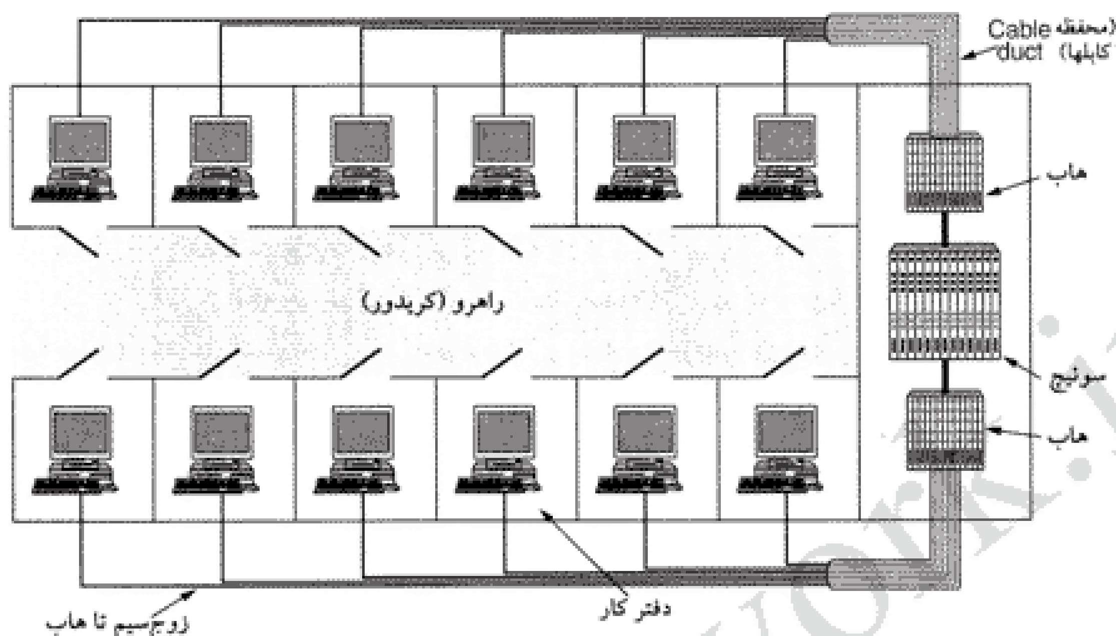
۴-۷-۶ شبکه‌های محلی مجازی (Virtual LANs)

در دوران اولیه به کارگیری شبکه‌های محلی، کابلهای زردرنگ ضخیم از طریق مجراهای مخصوص (duct)، بین دفاتر ساختمانها کشیده می‌شد و هر کامپیوتری که مسئولین امر تصمیم می‌گرفتند با وصل به این کابل، به شبکه ملحق می‌شد. اغلب، کابلهای بی شماری وجود داشت که به یک **«ستون فقرات مرکزی»** (Central Backbone) یا به یک هاب مرکزی متصل می‌شد (شکل ۴-۳۹) و این موضوع که کدام کامپیوتر به کدام LAN متصل می‌شود چندان مهم نبود. تمام افرادی که در دفاتر مجاور هم بودند بر روی یک LAN مشابه قرار می‌گرفتند، فارغ از آنکه کارشان ارتباطی با هم داشت یا نه! یعنی **«منطق برتر جغرافیائی»**. [بعبارت دیگر به جای آنکه یک منطق علمی، ساختار و اعضای یک LAN را تبیین کند جغرافیای محیط این ساختار و اعضاء را مشخص می‌کرد.]

با ابداع 10Base-T و هابها در اویل ۱۹۹۰ همه چیز عوض شد. همهٔ ساختمانها از نو سیم‌کشی شدند (البته با صرف هزینهٔ قابل توجه) تا تمام کابلهای زرد رنگ با قطر شیلنگ باغبانی برچیده شود و به جای آنها از هر دفتر یک زوج‌سیم به یک جعبه تقسیم مرکزی واقع در انتهای راهروها یا یک اتاق مرکزی کشیده شود. (شکل ۴-۴۸ را ببینید.) اگر معاون رئیس یا مسئول سیم‌کشی بلند پرواز بود سیم‌های زوجی ردهٔ ۵ (Cat 5) نصب می‌کرد ولی اگر تنگ نظر بود از سیم‌کشی موجود خطوط تلفن، استفاده می‌نمود (که باز هم چند سال بعد و با پدیدار شدن اترنت سریع باید عوض می‌شد!)

در اترنت مبتنی بر هاب (و بعداً مبتنی بر سوئیچ)، اغلب این امکان وجود داشت که LANها به جای پیکربندی جغرافیایی به صورت **منطقی** پیکربندی شوند. اگر شرکتی به k شبکهٔ LAN نیاز داشته باشد، k عدد هاب خریداری می‌کند. با دقت به آنکه کدام کابل رابط باید به کدام هاب متصل شود اعضای هر LAN را می‌توان به‌گونه‌ای انتخاب کرد که با ساختار سازمانی آنها مطابقت داشته باشد بدون آنکه جغرافیای محلّ، تاثیر چندانی در این انتخاب بگذارد. البته اگر دو نفر از یک دپارتمان مشابه از سازمان، در ساختمانهای متفاوتی مستقر بودند احتمالاً به دو هاب متفاوت و طبعاً به دو LAN متفاوت متصل می‌شدند. با این حال، شرایط جدید خیلی بهتر از زمانی است که اعضای یک LAN صرفاً بر اساس جغرافیای محلّ تعیین شوند.

آیا این مسئله که چه کسی برروی کدام شبکهٔ LAN قرار گرفته اهمیت دارد؟ مگر نه اینست که سرانجام در هر سازمان تمام LANها به هم متصل می‌شوند؟ کوتاه سخن آنکه جواب مثبت است و مسئله فوق اغلب مواقع اهمیت دارد. مسئولان شبکه بدلائل مختلف علاقمندند که کاربران را به نحوی برروی LAN گروه‌بندی کنند که گروه‌ها به جای آنکه منعکس کنندهٔ نقشه فیزیکی ساختمانها باشند، جلوه‌ای از ساختار سازمانی باشند. یکی از



شکل ۴ـ۴۸. یک ساختمان با سیم‌کشی مرکزی با بهره‌گیری از هاب و سوئیچ.

موارد و دلائل، «امنیت» است. هر کارت شبکه می‌تواند در حالت «بی‌قید» (Promiscuous) قرار بگیرد و تمام ترافیک جاری برروی کانال را دریافت نماید. بسیاری از دپارتمانها همانند دپارتمان پژوهش، ثبت و حسابداری، اطلاعاتی را در اختیار دارند که نمی‌خواهند به بیرون از دپارتمان خودشان راه پیدا کند. در چنین شرایطی قرار دادن تمام افراد برروی یک LAN واحد و جلوگیری از خروج ترافیک از آن LAN، معقول به نظر می‌رسد. مدیریت سازمان تمایلی به شنیدن آنکه «چنین آرایشی ممکن نیست» ندارند مگر آنکه تمام افراد هر دپارتمان، در دفاتر همجوار جای داده شده باشند و در کار یکدیگر فضولی نکنند!!

مورد دیگر «**میزان بار**» است: برخی از LANها نسبت به بقیه، زیادتر مورد استفاده قرار می‌گیرند و ممکن است که تفکیک آنها مطلوبتر باشد. به عنوان مثال اگر گروه پژوهش در حال اجرای انواع آزمایشاتی باشند که گاه ترافیکی بیش از اندازه تولید و شبکهٔ LAN را اشباع می‌کند، گروه حسابداری ممکن است علاقمند نباشد که برای کمک به آنان بخشی از ظرفیت [پهنای باند] خود را وقف آنان کند!

مورد سوّم، «پخش فراگیر» (Broadcasting) است. اغلب LANها از ارسال فراگیر حمایت می‌کنند و بسیاری از پروتکلهای لایه‌های بالاتر از این ویژگی در سطح گسترده‌ای استفاده می‌کنند. به عنوان مثال وقتی کاربری می‌خواهد بسته‌ای برای یک ماشین با آدرس IP معادل X بفرستد چگونه آدرس MAC آن ماشین را بدست می‌آورد تا در فریم مربوطه قرار بدهد؟ ما پاسخ این پرسش را در فصل پنجم بررسی خواهیم کرد ولی اگر بخواهیم بطور خلاصه جمع‌بندی نمائیم پاسخ آنست که ماشین فریمی را به صورت پخش فراگیر برروی شبکه قرار می‌دهد که حاوی این سوال است: «چه کسی صاحب آدرس IP معادل با X است؟»... سپس منتظر پاسخ باقی می‌ماند. نمونه‌های کاربردی زیادی می‌توان یافت که متکی به پخش فراگیر هستند. هر چه LANهای بیشتری به یکدیگر متصل شوند تعداد فریمهای فراگیر که به هر ماشین وارد می‌شوند به صورت خطی و متناسب با تعداد ماشینها افزایش خواهد یافت.

یکی دیگر از مشکلات پخش فراگیر آنست که اگر زمانی یک کارت شبکه از عملکرد طبیعی خود خارج شده و شروع به تولید جریان بی‌پایانی از فریمهای فراگیر نماید تکلیف چیست. نتیجهٔ به پا شدن این «**طوفان فریمهای**

فراگیر» آنست که (۱) کل ظرفیت LAN با این فریمها، اشغال و تباه می‌شود. (۲) تمام ماشینهای واقع در LANهای متصل به هم، به واسطهٔ صرف زمان جهت پردازش و سپس حذف این فریمهای فراگیر، زمین‌گیر می‌شوند.

در بدو امر ممکن است به نظر برسد که «طوفان فریمهای فراگیر» را می‌توان با جدا کردن LANها توسط پل یا سوئیچ محدود کرد ولی اگر هدف نهایی «شفافیت» باشد سوئیچها و پلها موظف به هدایت فریمهای پخش فراگیر هستند. (به عبارتی یک ماشین باید بتواند به یک LAN متفاوت تغییر موقعیت بدهد و هیچکس متوجه این موضوع نشود و در محل جدید نیز قادر به پخش فراگیر باشد.)

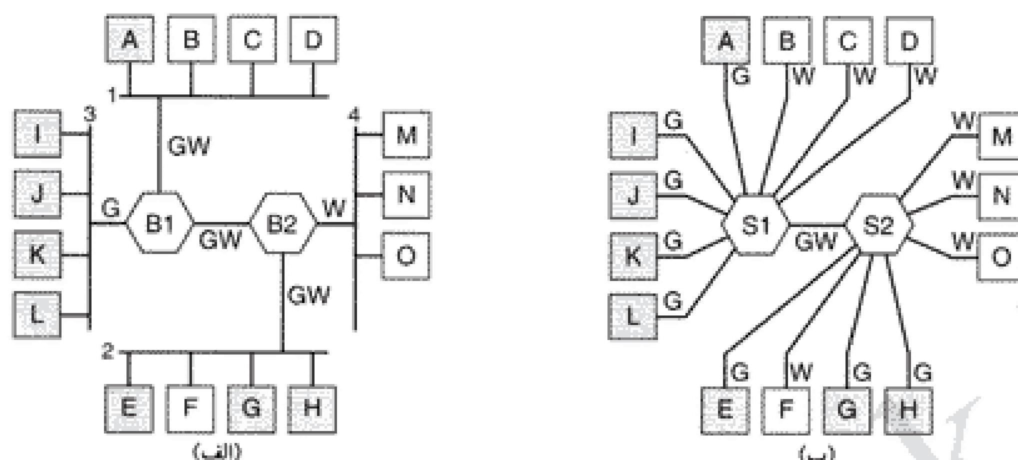
پس از آگاهی از آنکه چرا شرکتها ممکن است بخواهند LANهای متعددی با وسعت محدود داشته باشند، اجازه بدهید به مسئله اصلی یعنی جداکردن توپولوژی منطقی از توپولوژی فیزیکی بپردازیم. فرض کنید که یک کاربر در یک شرکت بدون تغییر محل دفتر کار خود از یک واحد اداری به واحد دیگر منتقل می‌گردد یا برعکس، دفتر کار خود را بدون جابجایی از واحد قبلی خود تغییر می‌دهد. در سیم‌کشی مبتنی بر هاب، تغییر موقعیت به یک LAN جدید مستلزم آنست که مسئول شبکه به سوی جعبه‌تقسیم (Wiring Closet) رفته و کابل رابط ماشین آن کاربر را از محل فعلی بیرون کشیده و آنرا به هاب جدید متصل کند.

در بسیاری از شرکتها تغییر و تحولات سازمانی یک امر عادی است و این مستلزم آنست که مسئول شبکه وقت بسیار زیادی را صرف جدا کردن کابلهای رابط و قرار دادن آن در محلّ جدید نماید. البته در برخی از حالات نیز چنین تغییراتی به هیچ وجه ممکن نیست چرا که مثلاً فاصلهٔ ماشین کاربر از هاب جدید بسیار دور است.

در پاسخ به نیاز کاربران به قابلیت انعطاف بیشتر، عرضه‌کنندگان محصولات شبکه کار برروی طرحی را آغاز کرده‌اند که بر اساس آن بتوان سیم‌کشی ساختمانها را به صورت «نرم‌افزاری» تغییر داد[یعنی پیکربندی ماشینهای هر LAN فارغ از ساختار اتصال فیزیکی آنها ممکن باشد.] نتیجهٔ چنین نظریه‌ای **«شبکهٔ محلّی مجازی»** یا VLAN نامیده می‌شود و توسط کمیته IEEE استانداردسازی شده است. اکنون بسیاری از سازمانها آرایشی مبتنی بر VLAN دارند. اجازه بدهید نگاهی به آن بیندازیم. برای کسب آگاهی بیشتر مراجع (Breyer and Riley , 1999, Seifert , 2000) مفیدند.

شبکه‌های VLAN، بکمک سوئیچهای خاصّ و سازگار با VLAN ، پیاده سازی می‌شوند، هر چند ممکن است همانند شکل ۴-۴۸ دارای چند هاب جانبی و معمولی نیز باشند. برای پیکربندی شبکهٔ مبتنی بر VLAN، مسئول شبکه تصمیم می‌گیرد که (۱) چند VLAN باید تعریف شود. (۲) چه کامپیوترهایی بر روی هر VLAN قرار می‌گیرند. (۳) هر یک از VLANها چگونه نامگذاری می‌شوند. اغلب، یک VLAN با رنگها نامگذاری می‌شود، (البته به صورت غیررسمی) زیرا بدین ترتیب امکان آنکه بتوان نقشه فیزیکی ماشینهای شبکه را به صورت رنگی چاپ کرد، وجود دارد. اعضای شبکهٔ LAN قرمز با رنگ قرمز مشخص می‌شوند، اعضای شبکه LAN سبز با رنگ سبز و به همین ترتیب. با این روش نقشهٔ منطقی و فیزیکی شبکه در یک نمای واحد دیده می‌شود.

به عنوان یک مثال، چهار LAN نشان داده شده در شکل ۴-۴۹ را مد نظر قرار بدهید که در آن هشت ماشین، متعلق به شبکه VLAN خاکستری (G) هستند، هفت تا متعلق به سفید (W) . چهار LAN فیزیکی نیز توسط دو پل B1 و B2 بهم متصل شده‌اند. اگر در این شکل به جای ساختار باس از هابهای مرکزی مبتنی بر زوج سیم استفاده می‌شد ممکن بود که چهار هاب نیز وجود داشته باشد (که در شکل نشان داده نشده است) ، ولیکن از دیدگاه منطقی کابلهای چنداتصالی و هاب مشابه به هم هستند. اگر می‌خواستیم شکل را با هاب ترسیم کنیم ، گویائی خود را از دست می‌داد. امروزه واژه پل بیشتر اشاره به ساختاری شبیه به شکل ۴-۴۹-ب دارد که در آن چندین ماشین از طریق پورتهای آن به هم متصل شده‌اند در حالیکه امروزه «پل» و «سوئیچ» مفهوم معادلی دارند. در شکل



شکل ۴-۴۹. (الف) چهار LAN فیزیکی با استفاده از دو پل، دو VLAN خاکستری و سفید تشکیل داده‌اند.(ب) همان پانزده ماشین بکمک دو سوئیچ، دو VLAN تشکیل داده‌اند.

۴-۴۹-ب همان تعداد ماشین و همان تعداد VLAN [معادل با شکل ۴-۴۹-الف] فقط با استفاده از سوئیچ نشان داده شده است بگونه‌ای که بر روی هر پورت سوئیچ فقط یک ماشین وجود دارد.

برای آنکه VLAN بدرستی کار کند بایستی یک جدول پیکربندی درون هر پل یا سوئیچ تنظیم شود. این جدول مشخص‌کنندهٔ آنست که از طریق کدام یک از پورتها می‌توان به کدام VLAN دسترسی داشت. وقتی فریمی از روی یک VLAN مثلاً خاکستری به سوئیچ وارد شود صرفاً باید بر روی تمام پورتهائی که علامت G دارند منتقل شوند. همانند ترافیک معمولی تک‌پخشی (Unicast) ، ارسال ترافیک چندپخشی (Muticast) و ارسال ترافیک فراگیر (Broadcast) نیز به همین نحو ممکن خواهد بود.

دقت کنید که ممکن است یک پورت برچسب رنگی چندین VLAN را داشته باشد. در شکل ۴-۴۹-الف چنین ساختاری را مشاهده می‌نمائیم. فرض کنید ماشین A یک فریم «فراگیر» [(Broadcast) را برای همهٔ اعضای VLAN خود] ارسال کند. پل B1 این فریم را دریافت کرده و متوجه می‌شود که توسط ماشینی بر روی VLAN خاکستری تولید شده است لذا آن فریم را برروی تمام پورتهائی که علامت G دارند (به استثنای پورتی که فریم از روی آن دریافت شده) ارسال می‌نماید. از آنجایی که B1 فقط دو پورت دیگر با برچسب G دارد لذا فریم را برروی هر دوی آنها ارسال می‌کند.

در B2 داستان به گونهٔ دیگری است: در اینجا پل می‌داند که هیچ ماشین خاکستری برروی LAN 4 وجود ندارد لذا فریم برروی آن هدایت نخواهد شد و فقط به LAN 2 ارسال می‌شود. اگر یکی از کاربران LAN 4 ملزم به تغییر واحد اداری خود شده و به VLAN خاکستری نقل مکان نماید باید جدول B2 بهنگام‌سازی شده و تنها پورت آن که بابرچسب W مشخص شده به GW تغییر داده شود. اگر ماشین F به شبکهٔ مجازی خاکستری نقل مکان کند، پورت متصل به LAN 2 باید از GW به G تغییر برچسب داده شود.

حال فرض کنید ماشینهای عضو LAN 2 و LAN 4 همگی به جمع خاکستری‌ها بپیوندند، لذا نه تنها پورتهای پل B2 که متصل به LAN 2 و LAN 4 هستند علامت G می‌گیرند بلکه پورت اتصال B1 به B2 نیز از GW به G تغییر علامت می‌دهد چراکه لازم نیست فریمهای سفیدی که از LAN 3 یا LAN 1 می‌رسند به B2 هدایت شوند. در شکل ۴-۴۹-ب همین وضعیت حاکم است و تمام پورتهایی که به یک ماشین واحد متصل هستند با یک برچسب تک‌رنگ مشخص می‌شوند چراکه هر ماشین تنها به یک VLAN متعلق است.

تاکنون فرض را بر آن گذاشته بودیم که پلها و سوئیچها به نحوی می‌دانند که رنگ یک فریم ورودی چیست.

آنها چگونه از این موضوع آگاه می‌شوند؟ برای این کار سه روش زیر کاربر دارد:

۱. به هر پورت رنگ VLAN انتساب داده شود.

۲. به هر آدرس MAC یک رنگ VLAN منتسب گردد.

۳. به آدرسهای لایهٔ ۳ یا آدرس IP ماشین یک رنگ VLAN منتسب شود.

در روش اوّل، به هر پورت یک برچسب رنگ داده می‌شود که این برچسب، VLAN مربوطه را مشخص می‌کند. با این حال این روش فقط زمانی کار خواهد کرد که تمام ماشینهای متصل به آن پورت، به یک VLAN مشابه متعلق باشند. در شکل ۴-۴۹-الف این ویژگی برای پورت بین پل B1 و LAN 3 صادق است در حالیکه برای پورت متصل به LAN 1 صادق نیست.

در روش دوّم، پل یا سوئیچ دارای جدولی است که در آن فهرست آدرسهای ۴۸ بیتی تمام ماشینهای متصل به آن (یعنی آدرس MAC) و مشخصات VLAN هر ماشین ، درج می‌شود. تحت این شرایط، می‌توان برروی یک LAN فیزیکی (مثل LAN 1 در شکل ۴-۴۹-الف) چند VLAN مختلط تعریف کرد. وقتی فریمی دریافت می‌شود، آنچه که هر پل یا سوئیچ باید انجام بدهد آنست که آدرس MAC آن را استخراج و درون جدول به دنبال آن بگردد و ببیند که فریم از کدام VLAN می‌آید.

در روش سوّم، پل یا سوئیچ موظف است تا محتوای فیلد حمل داده (Payload) از هر فریم را بررسی کرده و به عنوان مثال تمام ماشینهای مبتنی بر IP را در یک VLAN و ماشینهای مبتنی بر Apple Talk را در VLAN دیگر دسته‌بندی کند. در حالت اوّل می‌توان برای تشخیص هویت هر ماشین از آدرس IP آن استفاده کرد. این استراتژی زمانی بسیار سودمند خواهد بود که تعداد بی‌شماری از ماشینهای شبکه کامپیوترهای کیفی هستند و می‌توانند در هر یک از مکانهای متعدد و مجاز قرار بگیرند. از آنجایی که هر ایستگاه آدرس MAC خاص خود را دارد لذا با توقف ایستگاه در محل جدید، آدرس MAC آن نمی‌تواند چیزی در مورد آن VLAN که ایستگاه عضو آنست ، مشخص کند.

تنها مشکل این روش آنست که یکی از قوانین اساسی در شبکه بندی را نقض می‌کند: «استقلال لایه‌ها» تشخیص آنکه چه چیزی در فیلد حمل داده از فریم قرار گرفته برعهدهٔ لایهٔ پیوند داده‌ها نیست. این لایه نباید محتوای داده‌های درون هر فریم را بررسی کند و یا براساس محتوای درون آن تصمیمی بگیرد. یکی از تبعات به کارگیری این روش آنست که هرگاه در پروتکل لایهٔ ۳ تغییری ایجاد شود (مثلاً IPv4 به IPv6 ارتقاء پیدا کند) سوئیچ بلافاصله از کار خواهد افتاد. متاسفانه سوئیچهایی که بدین نحو کار می‌کنند به وفور در بازار عرضه شده‌اند.

البته تعریف VLAN بر اساس آدرس IP، اشکالی در مسیریابی بسته‌های IP بوجود نخواهد آورد ولیکن تلفیق وظائف لایه‌ها سرآغاز بروز مشکلات پیش‌بینی نشده خواهد بود. (تقریباً کلّ فصل پنجم به مسیریابی IP اختصاص دارد). شاید عرضه‌کنندگان سوئیچ چنین استدلالی را به مسخره بگیرند و بگویند سوئیچهای آنـه IPv4 و IPv6 را به رسمیت می‌شناسند و همه چیز به درستی پیش خواهد رفت ولیکن اگر زمانی IPv7 مطرح شد چه اتفاقی می‌افتد؟ شاید فروشنده بگوید که در آن زمان سوئیچهای جدید بخرید!! آیا این کار خیلی شاقّ است؟

استاندارد IEEE 802.1Q

اندکی اندیشهٔ بیشتر در خصوص VLAN ما را به این نتیجه می‌رساند که آنچه واقعاً اهمیت دارد آنست که هر فریم ارسالی نام VLAN خود را با خود حمل کند نه آنکه VLAN ماشین فرستنده را سوئیچ بروشهای دیگری مشخص کند. اگر روشی برای مشخص کردن VLAN در سرآیند هر فریم وجود داشته باشد دیگر نیازی به بررسی داده‌های

درونی هر فریم نخواهد بود. برای شبکه‌های جدیدی مثل 802.11 یا 802.16 ، اضافه کردن فیلد VLAN به سرآیند هر فریم نسبتاً ساده است. در حقیقت فیلد Connection Identifier در 802.16 ذاتاً چیزی شبیه به VLAN Identifier است. ولی در مورد اترنت چه کاری می‌توان کرد؟ شبکه‌ای که رایجترین شبکهٔ دنیاست و هیچ فیلد اضافی برای تعبیهٔ VLAN Identifier در آن، تعریف نشده است.

کمیتهٔ IEEE 802 این مسئله را در دستور کار خود قرار داد و پس از مباحثات فراوان کاری غیرقابل تصور انجام داد و سرآیند اترنت را عوض کرد. قالب جدید فریم، در استاندارد IEEE 802.1Q تدوین و در سال ۱۹۹۸ منتشر شد. در قالب جدید، هر فریم دارای یک برچسب VLAN است (با نام VLAN Tag) که مختصراً آن را بررسی خواهیم کرد. متاسفانه تغییر در استانداردی مثل سرآیند اترنت که کاملاً جا افتاده و بطور رایج از آن استفاده می‌شود، چندان ساده نیست. در این خصوص ممکن است سوالات زیر به ذهن خطور کند:

۱. آیا باید چندین میلیون کارت اترنت موجود را دور انداخت؟

۲. اگر نه، چه کسی فیلدهای جدید را تولید نماید؟

۳. برای فریمهایی که اندازهٔ حداکثر دارند چه اتفاقی می‌افتد؟ [چون نمی‌توان به فریمی که بر طول حداکثر آن محدودیت گذاشته شده، داده‌ای افزود.]

البته کمیتهٔ 802 از این مشکلات آگاه بود و می‌بایست راه حلّی مناسب ارائه می‌کرد، کاری که نهایتاً انجام شد.

نکتهٔ اصلی در راه حَل ارائه شده آنست که فیلدهای VLAN فقط توسط پلها و سوئیچها مورد استفاده قرار می‌گیرد و ماشینهای کاربران بدان نیازی ندارند. بدین ترتیب در شکل ۴-۴۹ وقتی فریمها مستقیماً به سوی یک ایستگاه پایانی ارسال می‌شوند به این فیلدها نیازی نیست و فقط روی خطوطی بین پلها یا سوئیچها ، بکار می‌آیند. بنابراین برای بکارگیری VLAN ، این پلها یا سوئیچها هستند که باید VLAN را به رسمیت بشناسند؛ این موضوع نیز در پلها و سوئیچها از قبل پیش‌بینی شده و یک نیاز تلقی می‌شود. حال باید نیازمندیهای جدیدی را معرفی نمائیم که 802.1Q بدانها پاسخ داده است.

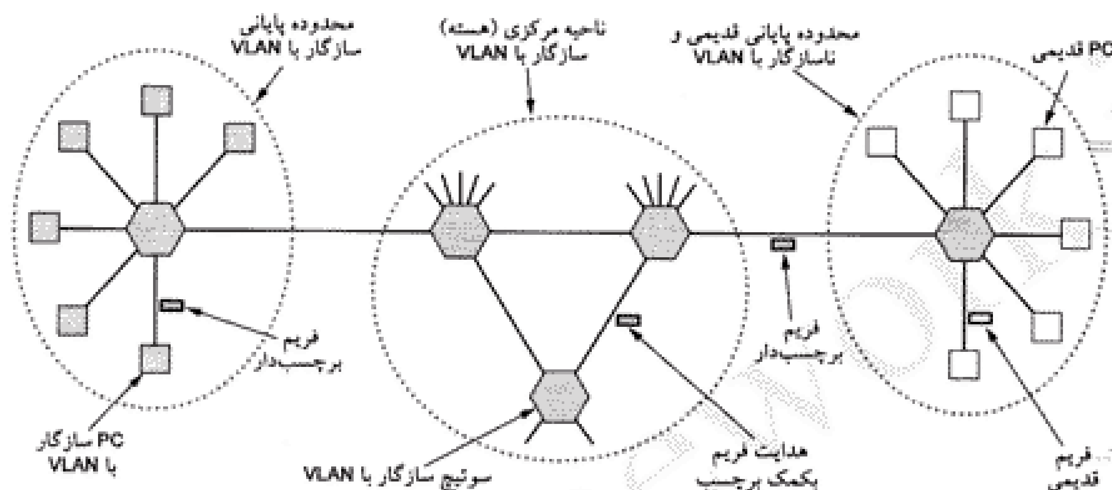
پاسخ به این سوال که «آیا باید تمام کارتهای اترنت موجود را دور انداخت؟» منفی است. به خاطر داشته باشید که کمیتهٔ 802.3 حتی نتوانست به افراد بقبولاند که فیلد Type را به فیلد Length تغییر بدهند. حال شما تصور کنید عکس‌العمل مردم در قبال اعلام دور انداختن کارتهای اترنت ، چه می‌شد! البته وقتی کارتها جدید به بازار بیاید شاید بتوان امیدوار بود که سازگار با 802.1Q بوده و فیلدهای مربوط به VLAN در آنها تعبیه شده باشند.

به هر حال اگر کارت شبکهٔ تولیدکنندهٔ یک فریم، فیلدهای VLAN را بدان اضافه نکند، چه کسی باید این کار را انجام بدهد؟ پاسخ این سوال آنست که اولین پل یا سوئیچ مبتنی بر VLAN که فریم را دریافت می‌کند، این فیلدها را به فریم افزوده و آخرین آنها در مسیر، این فیلدها را حذف می‌نماید. ولیکن یک سوئیچ یا پل از کجا می‌فهمد که یک فریم به کدام VLAN تعلق دارد؟ پاسخ آنست که اولین پل یا سوئیچ می‌تواند به هر یک از پورتهای خود یک شمارهٔ VLAN نسبت بدهد یا آنکه به آدرس MAC آن فریم نگاه کند و یا محتوای داده‌های درونی را بررسی کند (کار ممنوع!).

در خصوص کارتهای اترنت که سازگار با 802.1Q هستند، مشکلی وجود ندارد. انتظار قابل تحقق آنست که تمام کارتهای اترنت گیگابیت از همان ابتدا سازگار با 802.1Q باشند و با ارتقاء کارتهای قدیمی به اترنت گیگابیت ، 802.1Q نیز به صورت خودکار جای خود را باز کند. برای حلّ این مشکل که فریمهای اترنت نباید از 1518 بایت بیشتر باشند در 802.1Q طول حداکثر به ۱۵۲۲ بایت افزایش یافته است.

در دوران گذار از اترنت فعلی به اترنت گیگابیت، در بسیاری از شبکه‌ها برخی از ماشینهای قدیمی (عموماً

اترنت کلاسیک و اترنت سریع) با VLAN سازگار نیستند، در حالیکه برخی دیگر (عموماً اترنت گیگابیت) از آن پشتیبانی می‌کنند. چنین وضعیتی در شکل ۴-۵۰ به تصویر کشیده شده است و در آن نمادهای خاکستری سازگار با VLAN و نمادهای بی‌رنگ ناسازگار هستند. برای سادگی بحث، فرض را بر آن گذاشته‌ایم که تمام سوئیچها با VLAN سازگار هستند. اگر اینگونه نباشد اوّلین سوئیچ سازگار با VLAN بکمک آدرس MAC یا آدرس IP درون فریم، برچسب لازم را بدان خواهد افزود.



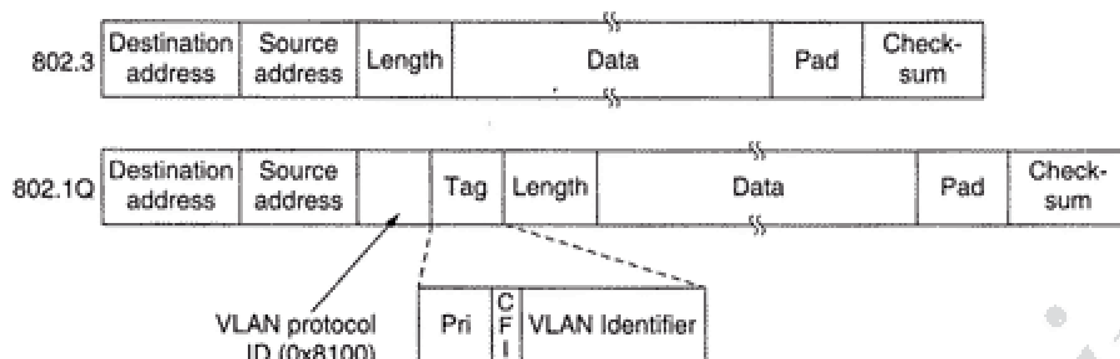
شکل ۴-۵۰. گذر از اترنت قدیمی به اترنت سازگار با VLAN. نمادهای خاکستری با VLAN سازگارند؛ نمادهای بی‌رنگ سازگار نیستند.

در این شکل کارتهای اترنت سازگار با VLAN ، مستقیماً فریمهای برچسب‌دار VLAN (یعنی فریمهای 802.1Q) را تولید می‌نمایند و سوئیچهای بعدی از این برچسب استفاده خواهند کرد. برای عملیات هدایت فریمها (یعنی عمل سوئیچنگ) هر سوئیچ بایستی بداند که کدام VLAN از طریق کدام پورت در دسترس است. (طبق توضیحات قبلی) دانستن آنکه فریم جاری به VLAN مثلاً خاکستری تعلق دارد فایدهٔ چندانی نخواهد داشت مگر آنکه سوئیچ بداند کدام پورتها به ماشینهای عضو VLAN خاکستری متصل هستند. بنابراین سوئیچ نیازمند به داشتن جدولی اندیس‌دار و مرتب شده بر حسب مشخصهٔ VLAN است که تعیین می‌کند از کدامیک از پورتها باید استفاده شود و آیا ماشینهای متصل بدین پورتها سازگار با VLAN هستند یا قدیمیند.

وقتی یک PC قدیمی فریمی را برای یک سوئیچ سازگار با VLAN می‌فرستند، سوئیچ مربوطه با استفاده از دانش قبلی[1] خود در خصوص VLAN متناظر با فرستندهٔ فریم (مثلاً با استناد به شمارهٔ پورت فیزیکی، آدرس MAC یا آدرس IP) برچسب لازم را به فریم چسبانده و فریم جدیدی تولید می‌نماید. از این دیدگاه، دیگر قدیمی بودن ماشین فرستنده هیچ اهمیتی ندارد. به روش مشابه هرگاه نیاز باشد فریمی برچسب‌دار (مبتنی بر 802.1Q) تحویل ماشینی قدیمی شود قبل از تحویل، ساختار فریم به قالب بدون برچسب تبدیل خواهد شد.

حال بیائید نگاهی به قالب فریم 802.1Q بیندازیم. قالب این نوع فریم در شکل ۴-۵۱ نشان داده شده است تنها تغییر، اضافه شدن یک جفت فیلد ۲ بایتی (جمعاً ۴ بایت) به قالب قدیمی است. یکی از آنها فیلد مشخصه VLAN Protocol ID است که همیشه مقدار 0x8100 دارد. از آنجایی که این مقدار بیش از ۱۵۰۰ است تمام

۱. به خاطر داشته باشید که سوئیچهای مدرن سازگار با VLAN ، بصورت نرم‌افزاری پیکربندی می‌شوند و بعضاً دارای سیستم عامل و فرامین مخصوص هستند. بنابراین در خصوص ماشینهای باکارت اترنت قدیمی تنظیمات VLAN ، باید بصورت دستی انجام شود.-م



شکل ۴ـ۵۱. قالب فریم قدیمی اترنت 802.3 و فریم 802.1Q.

کارتهای اترنت آنرا به عنوان فیلد «نوع فریم» (Type) تفسیر می‌کنند و به فیلد طول داده Length تعبیر نخواهد شد. عکس‌العملی که کارتهای اترنت قدیمی در برخورد با چنین فریمهایی از خود نشان می‌دهند جای بحث دارد چراکه فرض بر آن است که چنین فریمهایی برای کارتهای قدیمی ارسال نمی‌شود.

فیلد دو بایتی بعدی شامل سه زیرفیلد است که اصلی‌ترین آنها یعنی VLAN Identifier (شناسهٔ VLAN)، ۱۲ بیت کم‌ارزش را به خود اختصاص داده است. این فیلد کل آن چیزی است بدان نیازمند بوده‌ایم: یعنی مشخص می‌کند که «این فریم به کدام VLAN تعلق دارد». فیلد ۳ بیتی Priority (اولویت) فعلاً کاری در خصوص VLAN بر عهدهٔ ندارد و عملاً بلااستفاده است. استدلال کمیتهٔ 802.1Q در تعریف این زیرفیلد آن بوده که چون تغییر در سرآیند اترنت فرآیند دشوار و زمانبری است، حالا که این کار خطیر میسر شده چرا چیزهای خوب دیگر نیز بدان اضافه نشود؟ این فیلد ۳ بیتی این امکان را فراهم آورده تا بتوان «ترافیک بی‌درنگ از نوع سخت و نرم» (Soft & Hard Realtime Traffic) [مثل صدا و تصویر یا نظائر آن] را از ترافیک غیرحساس به زمان [مثل ترافیک ارسال نامه‌های الکترونیکی] تشخیص داد و امکان ارائه «کیفیت خدمات» (QoS) در اترنت نیز مسیر شود. در آینده به این فیلد جهت ارسال صدا بر روی اترنت نیاز خواهد شد. (هر چند مشابه با همین فیلد در پروتکل IP ربع قرن قبل در نظر گرفته شده بود ولیکن هیچگاه از آن استفاده نشد!!)

آخـرین زیـرفیلد، بیت CFI (Canonical Format Indicator) است کـه در حـقیقت مـی‌بایست بیت CEI (Corporate Ego Indicator) بمعنای مشخصهٔ شخصی شرکت نامیده می‌شد. اساساً این بیت باید برای مشخص کردن آدرسهای Big Endian از آدرسهای Little Endian بکار مـی‌رفت ولی پس از بـحث و جـدل فراوان از آن صرفنظر شد.[1] اکنون معنای این بیت آن است که درون فیلد حمل داده از فریم اترنت، یک فریم کامل از نوع 802.5 (فریم شبکهٔ Token Ring) قرار دارد و انتظار می‌رود به یک شبکهٔ محلی از نـوع 802.5 دیگـر تحویل شود چراکه اتصال آنها از طریق اترنت برقرار شده است. (به عبارت دیگر این فیلد مشخص می‌کند که شبکهٔ اترنت باید به عنوان حامل میانی، یک فریم نوع 802.5 را به شبکه‌ای دیگر از همین نوع تحویل بدهد.)[2] البته این بیت هیچ کاری در مورد شبکهٔ VLAN انجام نمی‌دهد ولیکن سیاستهای کمیته استانداردسازی IEEE متفاوت با سیاستهای عادّی است!!!

با دقت در توضیحات فوق، وقتی یک فریم برچسب‌دار به یک سوئیچ سازگار با VLAN می‌رسد آن سوئیچ از

---

۱. معنای آدرس Big Endian آن است که بایتهای پرارزش آدرس در ابتدا و بایتهای کم‌ارزش در انتها قرار گرفته و به همان نحو ارسال می‌شوند. معنای آدرس Little Endian آن است که بایتهای کم‌ارزش در ابتدا و بایتهای پر ارزش آدرس بعد از آن ارسال می‌شوند.

۲. یعنی نوعی از پدیدهٔ تونل‌زنی (Tunneling) در سطح لایهٔ پیوند داده. ـم

فیلد VLAN ID به عنوان اندیس (یا بعبارتی کلید جستجو) در جدول خود استفاده کرده و به کمک آن پورتهایی که این فریم باید بر روی آنها فرستاده شود را مشخص می‌کند. ولی سوال اساسی اینجاست که این جدول از کجا بدست می‌آید؟ اگر این جدول باید به صورت دستی تنظیم شود به پلهٔ شمارهٔ صفر برگشته‌ایم یعنی «پیکربندی دستی پلها» ؛ زیبائی پلهای شفاف در آن است که به محض وصل ، فوراً به کار می‌افتند (یعنی Plug & Play است) و نیازی به پیکربندی دستی ندارند. از دست دادن این ویژگی بسیار ناگوار است! خوشبختانه پلهای سازگار با VLAN می‌توانند خود را بر اساس برچسب فریمهای ورودی به صورت خودکار پیکربندی کنند. مثلا اگر فریمی با برچسب VLAN 4 از روی پورت شماره ۳ وارد شود ، بهٔ روشنی مشخص است که ماشین متصل به پورت ۳ متعلق به VLAN 4 است. استاندارد 802.1Q روش ایجاد پویای این جداول را تشریح کرده ولی در اکثر جاهای این توضیحات خواننده را به بخشهایی خاصّ از الگوریتم پرلمن ارجاع داده است؛ الگوریتم پرلمن نیز در 802.1D استاندارد شده است.

قبل از خاتمهٔ بحث مسیریابی در VLAN ، اشاره به یک نکتهٔ پایانی خالی از لطف نیست. بسیاری از افراد در دنیای اینترنت و اترنت طرفدار ساختار شبکه‌های «بی‌اتصال» (Connectionless) هستند و در مقابل هر چیزی که زمینه نیاز به «اتصال» در لایهٔ «پیوند داده» یا لایهٔ شبکه را فراهم کند به شدت مقاومت می‌کنند. در VLAN چیزهایی معرفی شده که شبیه به مفهوم «اتصال» هستند. برای عملکرد صحیح VLAN هر فریم با خود یک مشخصه خاص و جدید دارد [یعنی همان VLAN ID] که از آن بعنوان اندیس جستجو در جدول درون سوئیچ ، استفاده می‌شود. این دقیقاً همان اتفاقی است که در شبکه‌های اتصال‌گرا می‌افتد. در شبکه‌های بدون‌اتصال به تنها چیزی که برای مسیریابی و هدایت فریم نیاز است آدرس MAC می‌باشد و به هیچ نوع مشخصهٔ اتصال یا امثال آن نیاز نیست. در فصل پنجم در خصوص مفاهیم اتصال‌گرائی (Connectionism) بیشتر خواهیم آموخت.

۴ـ۸ خلاصه

برخی از شبکه‌ها تنها دارای یک کانال انتقال مشترک هستند که همهٔ ایستگاهها از آن، برای مبادله داده‌های خود استفاده می‌کنند. در این شبکه‌ها، مسئله اصلی در طراحی لایهٔ پیوند داده ، تسهیم و تخصیص کانال بین ایستگاه‌های رقیبی است که علاقمندند از آن استفاده نمایند. الگوریتمهای بی‌شماری برای مسئله تخصیص کانال ابداع شده است. خلاصه‌ای از مهمترین روشهای تخصیص کانال در جدول ۴-۵۲ فهرست شده است.

ساده‌ترین روش تخصیص، FDM و TDM است. این روشها زمانی مفیدند که تعداد ایستگاه‌ها کم و ثابت و ترافیک آنها پیوسته باشد. هر دوی این روشها در محیطهایی با این ویژگی مثلاً برای تقسیم پهنای باند در خطوط اصلی تلفن (شاهراه‌ها)، کاربرد دارند. وقتی تعداد ایستگاه‌ها زیاد و متغیر یا ترافیک آنها نسبتاً انفجاری است روش‌های TDM و FDM گزینه‌های مناسبی نیستند. پروتکل ALOHA (چه نوع معمولی و چه نوع Slotted آن) بعنوان روشهای جایگزین TDM و FDM معرفی شدند. گونه‌های مختلف ALOHA نیز تشریح و تحلیل شده‌اند. برخی از این گونه‌ها امروزه در سیستمهای واقعی به کار می‌روند.

وقتی بتوان حالت فعلی کانال را شنود (احساس) کرد، ایستگاه‌ها قادر خواهند بود مادامیکه ایستگاه دیگر در حال ارسال است از ارسال فریم خود اجتناب کنند. این تکنیک یعنی «**شنود سیگنال حامل**» منتج به ابداع پروتکلهای متنوعی شد که می‌تواند در شبکه‌های LAN و MAN به کار گرفته شود.

رده‌ای از پروتکلها قادرند مسئله رقابت بر سر تصرف کانال را منتفی کنند یا حداقل این رقابت را تا حّد قابل توجهی کاهش بدهند. روش «شمارش دودوئی معکوس» (Binary Countdown) رقابت را بطور کامل حذف می‌کند. پروتکل «پیمایش درخت وفقی» (Adaptive Tree Walk) با تقسیم‌بندی پویای ایستگاه‌ها به گروه‌های از

روش	توصیف عملکرد
FDM	به هر ایستگاه یک باند فرکانسی تخصیص می‌دهد.
WDM	الگوی پویای FDM برای فیبر نوری
TDM	به هر ایستگاه یک برش زمان تخصیص می‌دهد.
Pure ALOHA	ارسال ناهماهنگ در هر لحظه دلخواه
Slotted ALOHA	ارسال تصادفی در برشهای زمانی مشخص
1-persistent CSMA	استاندارد دسترسی چندگانه مبتنی بر شنود کانال
Nonpersistent CSMA	تاخیر تصادفی در هنگامی که کانال اشغال تشخیص داده می‌شود.
P-persistent CSMA	همان CSMA است با این تفاوت که احتمال اصرار در ارسال فریم P است.
CSMA/CD	همان CSMA است با این تفاوت که بمحض تشخیص تصادم ادامه ارسال را قطع می‌کند.
Bit map	زمان‌بندی به نوبت و چرخشی (Round-Robin) بکمک یک الگوی بیتی (Bitmap)
Binary countdown	ایستگاه با بزرگترین شماره حق ارسال در نوبت بعدی را دارد.
Tree walk	رقابت محدود با فعالیتهای انتخابی (تعریف گروه‌های رقیب در ساختار درختی)
MACA, MACAW	پروتکلهای شبکه‌های محلی بی‌سیم
Ethernet	همان CSMA/CD با الگوریتم عقبگرد نمایی
FHSS	جهشهای فرکانسی در طیف گسترده (Frequency Hopping Spread Spectrum)
DSSS	Direct Sequence Spread Spectrum
CSMA/CA	استاندارد دسترسی چندگانه مبتنی بر شنود کانال و اجتناب از تصادم

شکل ۴-۵۲. روشها و سیستمهای تخصیص یک کانال مشترک.

هم جدا، مشکل رقابت را بطور جدّی کاهش می‌دهد. تقسیم‌بندی گروه‌ها حتی‌الامکان به نحوی است که فقط گروههایی که در آنها فقط یک ایستگاه فریمی برای ارسال آماده دارد مجاز به ارسال هستند.

شبکه‌های محلی بی‌سیم مشکلات و راه حلّ‌های خاص خود را دارند. عمده‌ترین مشکل توسط ایستگاه‌های «پنهان» ایجاد می‌شود و بدین دلیل CSMA جوابگوی چنین شبکه‌هایی نخواهد بود. رده‌ای از این راه‌حلها که در طبقهٔ MACA و MACAW دسته بندی شده‌اند سعی دارند ایستگاه‌ها را وادار به ارسال در پیرامون ماشین مقصد نمایند تا بدین نحو CSMA عملکرد بهتری داشته باشد. روش‌های مبتنی بر طیف گسترده از نوع Frequency Hopping و Direct Sequence نیز در این شبکه‌ها به کار گرفته شده است. IEEE 802.11 روش CSMA و MACAW را تلفیق کرد و روش CSMA/CA را معرفی نمود.

اترنت رایجترین نوع شبکه‌های محلی است که برای تخصیص کانال از روش CSMA/CD بهره می‌گیرد. نسخه قدیمی آن از یک کابل واحد که بین تمام ماشینها کشیده می‌شد، استفاده می‌کردند، در حالیکه اکنون استفاده از هاب، سوئیچ و کابلهای زوجی بسیار رایج است. سرعت آن از ۱۰ مگابیت‌برثانیه تا یک گیگابیت بر ثانیه افزایش یافته و هنوز هم در حال افزایش است.

امروز شبکه‌های محلی بی‌سیم نیز در حال رواج هستند که در این بین استفاده از 802.11 وسیعتر است. لایهٔ فیزیکی در این شبکه، امکان پنج نوع انتقال و مدولاسیون متفاوت را فراهم آورده است که شامل مادون قرمز، گونه‌های متفاوت مبتنی بر طیف گسترده و «سیستم FDM چندکاناله» است. این شبکه می‌تواند یک ایستگاه ثابت در هر سلول داشته باشد ولیکن قادر است بدون ایستگاه ثابت نیز کار کند. این پروتکل گونه‌ای از MACAW و متکی بر شنود مجازی سیگنال حامل است.

شبکه‌های بین شهری بی‌سیم (Wireless MAN) در حال پیدایش و رواج هستند. این گونه شبکه‌ها

سیستمهای باند گسترده‌ای هستند که از امواج رادیوئی بهره می‌گیرند تا جایگزین ارتباطات تلفنی شوند. در آنها از تکنیک‌های مدولاسیون باند باریک استفاده شده است. کیفیت خدمات نیز از موارد مهمی است که استاندارد 802.16 چهار ردهٔ مختلف از این گونه خدمات را تعریف کرده است: ارسال با نرخ ثابت، دو روش ارسال با نرخ متغیر و روش ارسال مبتنی بر بهترین تلاش (Best Effort).

سیستم بلوتوث نیز بی‌سیم است ولیکن هدف آن وصل ابزارهای رومیزی است؛ مثلا برای اتصال گوشیها و ابزارهای جانبی کامپیوترهای شخصی بدون نیاز به سیم کاربرد دارد. همچنین به منظور وصل ابزارهایی مثل دورنگار (فکس) و تلفن‌های همراه به کار می‌آید. بلوتوث شبیه به 802.11 از تکنیک طیف گسترده مبتنی بر پرش فرکانس بهره گرفته و درباند ISM [یعنی 2.4 Ghz] کار می‌کند. به خاطر بالا بودن سطح نویز در بسیاری از محیطها و نیاز به فعل و انفعال بی‌درنگ، در پروتکلهای مختلف آن از روش تصحیح خطای مستقیم (Forward Error Correction) بهره گرفته شده است.

بدلیل وجود انواع متفاوت LAN به روشی جهت اتصال آنها به یکدیگر نیاز است. از پل و سوئیچ به همین منظور استفاده می‌شود. در پلهای نوع Plug & Play از «الگوریتم درخت پوشا» (Spanning Tree) استفاده شده است. پیشرفت جدید در دنیای شبکه‌ها، VLAN است که بکمک آن توپولوژی منطقی LANها از توپولوژی فیزیکی آن جدا می‌شود. قالب جدیدی برای فریمهای اترنت معرفی شده است تا راحتر بتوان VLAN را در سازمان‌ها پیاده سازی کرد.

مسائل

۱. برای حل این مسئله از رابطه‌ای که در همین فصل آمده بهره بگیرید ولیکن در ابتدا آن را بیان نمائید. بطور تصادفی فریمهایی برای انتقال بر روی کانالی با نرخ ارسال 100 Mbps تولید می‌شوند. اگر در زمان تولید یک فریم کانال اشغال بود تا رسیدن نوبت به آن در صف منتظر خواهد ماند. طول فریمها دارای تابع توزیع نمائی با میانگین 10000 bits/Frame است. برای هر یک از نرخهای تولید فریم که در زیر مشخص شده، تاخیری را که هر فریم (با طول متوسط) با آن مواجه می‌شود (شامل زمان انتظار صفّ و زمان انتقال) حساب نمائید.

الف) ۹۰ فریم در ثانیه
ب) ۹۰۰ فریم در ثانیه
ج) ۹۰۰۰ فریم در ثانیه

۲. یک گروه Nتائی از ایستگاه‌ها دارای کانالی مشترک با نرخ 56kbps هستند و روش تخصیص کانال نیز Pure ALOHA است. بطور متوسط هر ایستگاه در هر صد ثانیه یک فریم هزار بیتی تولید می‌کند (حتی اگر فریم قبلی هنوز ارسال نشده باشد چراکه ایستگاه‌ها می‌توانند فریمهای خروجی را در بافر خود نگاه دارند). مقدار حداکثر N چقدر می‌تواند باشد؟

۳. تاخیر Pure ALOHA را در مقایسه با روش Slotted ALOHA و در شرایط بار پائین مدّ نظر قرار بدهید. تاخیر کدامیک کمتر است؟ پاسخ خود را تشریح کنید.

٤. ده هزار ایستگاه رزرو بلیط هواپیما، برای استفاده از یک کانال واحد، بروش Slotted ALOHA با هم رقابت می‌کنند. هر ایستگاه بطور متوسط ۱۸ تقاضا در هر ساعت خواهد داشت. برشهای زمانی (Time Slot) ۱۲۵ میکروثانیه‌ای هستند. مقدار تقریبی بار کّل کانال چقدر است؟

۵. جمع کثیری از کاربران سیستم ALOHA در هر ثانیه ۵۰ تقاضا تولید می‌نمایند (که این مقدار شامل تعداد

تقاضاهای جدید و تقاضاهای ارسال مجدد فریمهایی است که قبلا در اثر تصادم خراب شده‌اند). زمان به برشهای متساوی ۴۰میلی‌ثانیه‌ای تقسیم شده است:

الف) احتمال موفقیت ارسال در همان دفعه اول چقدر است؟

ب) احتمال بروز دقیقاً k تصادم پیاپی و سپس یک موفقیت چقدر است؟

ج) به طور متوسط برای ارسال یک فریم چند بار تلاش لازم است؟

۶. اندازه‌گیری بار کانال Slotted ALOHA با تعداد نامحدود کاربر، نشان می‌دهد که ده درصد از برشهای زمانی بلااستفاده مانده‌اند:

الف) بار کانال یعنی G چقدر است؟

ب) بازده مفید کانال (Throughput) چقدر است؟

ج) آیا بار کانال بیش از اندازه بالاست یا کمتر از حد متعادل است؟

۷. در یک سیستم Slotted ALOHA با کاربران نامحدود، متوسط تعداد برش زمانی که هر ایستگاه بین تصادم و ارسال مجدّد صبر می‌کند، ۴ است. منحنی «تاخیر» را بر حسب «بازده مفید کانال» (Delay Versus throughput) در این سیستم ترسیم نمائید.

۸. یک ایستگاه مثل S را در شبکه‌ای با پروتکلهای زیر در نظر بگیرید. این ایستگاه قبل از ارسال فریمش بر روی کانال ، (در بدترین حالت) چه مدت زمانی باید انتظار بکشد:

الف) پروتکل Basic Bit Map

ب) پروتکلهای Mok و Wark با جابگشت مجازی شمارهٔ ایستگاه‌ها

۹. یک LAN از پروتکل «شمارش دودوئی معکوس» نسخهٔ Mok و Wark ، بهره می‌گیرد. در یک لحظهٔ خاص، ده ایستگاه دارای شماره‌های مجازی ۸، ۲، ۴، ۵، ۱، ۷، ۳، ۶، ۹ و ۰ هستند. سه ایستگاه ارسال‌کنندهٔ بعدي به ترتیب عبارتند از: ۴، ۳ و ۹. پس از آنکه این ایستگاه‌ها ارسال خود را به اتمام رساندند شماره مجازی ایستگاه‌ها چند خواهد بود؟ [ترتیب شماره‌ها را از راست به چپ در نظر بگیرید.]

۱۰. شانزده ایستگاه که از ۱ تا ۱۶ شماره‌گذاری شده‌اند برای استفاده از یک کانال اشتراکی بروش «پیمایش درخت وفقی» رقابت می‌کنند. اگر تمام ایستگاه‌هایی که آدرس آنها اعداد اول است بطور ناگهانی آماده ارسال شوند برای خاتمه مراحل رقابت چند «برش بیتی» (Bit Slots) نیاز خواهد بود؟

۱۱. مجموعه‌ای از 2^n ایستگاه برای دسترسی به یک کابل اشتراکی از روش «پیمایش درخت وفقی» بهره گرفته‌اند. در یک زمان خاصّ دو ایستگاه آماده ارسال می‌شوند. اگر $2^n>>1$ باشد مقدار حداقل، حداکثر و میانگین تعداد برشهای زمانی برای پیمایش این درخت [و خاتمهٔ رقابت] چقدر است؟

۱۲. در شبکهٔ محلی بی‌سیم که مطالعه کردیم به جای استفاده از پروتکل CSMA/CD از پروتکلی نظیر MACA استفاده شده بود. تحت چه شرایطی (در صورت امکان) استفاده از روش CSMA/CD میسر خواهد بود؟

۱۳. چه ویژگیهایی از پروتکلهای دسترسی به کانال در GSM و WDMA مشترک هستند؟ GSM را در فصل ۲ مرور نمائید.

۱۴. شش ایستگاه A تا F با بکارگیری پروتکل MACA با یکدیگر در ارتباط و تبادل داده هستند. آیا امکان انتقال همزمان اطلاعات در این شبکه وجود دارد؟ (پاسخ خود را شرح بدهید.)

۱۵. در هر طبقه از یک ساختمان هفت طبقه، ۱۵ دفتر کار همجوار وجود دارد. در هر دفتر یک پریز دیواری (Wall Socket) برای وصل یک پایانه در جلوی آن تعبیه شده است و بدین ترتیب پریزها یک مستطیل در چهارگوش یک صفحه قائم تشکیل داده‌اند که فاصله بین پریزها به صورت عمودی و افقی چهار متر است.

فرض کنید امکان کابل‌کشی مستقیم بین هر جفت پریز، چه عمودی، افقی یا مورب وجود داشته باشد. برای اتصال این پریزها به هم، به چند متر کابل نیاز است اگر:

الف) پیکربندی مبتنی بر «ستاره» و با یک مسیریاب در وسط، مدنظر باشد.

ب) شبکهٔ محلی 802.3 مدنظر باشد.

۱۶. نرخ تغییرات سیگنال (یعنی Baud Rate) در استاندارد ده‌مگابیتی اترنت چقدر است؟

۱۷. سیگنال حاصل از کدینگ «منچستر» را برای رشته بیت 0001110101 نشان بدهید.

۱۸. سیگنال حاصل از کدینگ «منچستر تفاضلی» را برای رشته بیت مسئله بالا ترسیم نمائید. فرض کنید خط در هنگام شروع به ارسال در حالت LOW قرار داشته باشد.

۱۹. در یک شبکهٔ محلی 10Mbps مبتنی بر CSMA/CD (البته نه 801.3) با طول یک کیلومتر، سرعت انتشار سیگنال ۲۰۰ متر بر میکروثانیه است. طول فریمهای داده ۲۵۶ بیت شامل مجموعاً ۳۲ بیت سرآیند، کد کشف خطا و سربارهای دیگر است. اولین برش بیتی (Bit Slot) پس از ارسال موفق، برای گیرندهٔ فریم در نظر گرفته شده تا بتواند برای ارسال پیام ACK (تصدیق دریافت فریم) کانال را در اختیار بگیرد. نرخ ارسال مفید (بدون در نظر گرفتن سربار و با فرض عدم وجود تصادم) چقدر است؟

۲۰. دو ایستگاه CSMA/CD تلاش می‌کند تا یک فایل بزرگ (شامل چندین فریم) را ارسال نمایند. پس از ارسال هر فریم، آنها برای دسترسی به کانال طبق الگوریتم «عقب‌گرد نمایی» با یکدیگر رقابت می‌کنند. احتمال اینکه هر رقابت در k دور به اتمام برسد و همچنین تعداد متوسط دورها (rounds) در هر رقابت چقدر است؟

۲۱. ساختمانی را در نظر بگیرید که در آن شبکه‌ای CSMA/CD با سرعت 1Gbps و یک کیلومتر کابل، بدون هیچ تکرارکننده (Repeater) کار می‌کند. سرعت انتشار سیگنال روی کابل ۲۰۰۰۰۰ کیلومتر بر ثانیه است. مقدار حداقل طول هر فریم چقدر است؟

۲۲. طول یک بسته IP جهت ارسال بر روی اترنت، ۶۰ بایت (شامل کل سرآیندها) است. اگر از LLC استفاده نشده باشد آیا در فریم اترنت به عمل اضافه کردن داده‌های زائد (Padding) نیاز است و اگر نیاز است چند بایت؟

۲۳. طول فریمهای اترنت باید حداقل ۶۴ بایت باشد تا این اطمینان حاصل شود که در صورت تصادم در انتهای کابل، فرستنده کماکان در حال ارسال است.[تا بتواند قبل از خاتمهٔ فریم تصادم را کشف کند.] حداقل طول فریمها در «اترنت سریع»، ۶۴ بایت است در حالی که ارسال بیتها برروی خروجی ۱۰ برابر سریعتر شده است. چگونه ممکن بوده که بتوان طول حداقل فریمها را در همان ۶۴ بایت نگه داشت؟

۲۴. در برخی از کتابها طول حداکثر هر فریم اترنت ۱۵۱۸ بایت ذکر شده است. آیا اشتباهی در کار است؟ پاسخ خود را تشریح نمایید.

۲۵. در تشریح مشخصات 1000Base-SX بیان شده که سیگنال ساعت در فرکانس ۱۲۵۰ MHz کار می‌کند، در حالیکه در اترنت گیگابیت فرض بر آن است که نرخ تحویل داده‌ها 1Gbps است. آیا این سرعت بالاتر، برای ایجاد حاشیهٔ اطمینان بیشتر بوده است؟ اگر نه، چه نکته‌ای در کار است؟

۲۶. اترنت گیگابیت از عهدهٔ ارسال یا دریافت چند فریم در هر ثانیه بر می‌آید؟ به دقت بیندیشید و تمام حالات را در نظر بگیرید. راهنمایی: واقعیاتی که در شبکهٔ اترنت گیگابیتی وجود دارد را مدّ نظر قرار بدهید.

۲۷. دو شبکه را نام ببرید که اجازه می‌دهند فریمها بطور پشت سرهم و در یک بار ارسال انتقال یابند. به چه دلیل چنین خصوصیتی ارزشمند است؟

۲۸. در شکل ۴-۲۷ چهار ایستگاه A، B، C و D نشان داده شده‌اند. به نظر شما کدام یک از ایستگاه‌ها به A نزدیکتر هستند؟

۲۹. فرض کنید که در یک شبکه محلی بی‌سیم 802.11b [پس از در اختیار گرفتن کانال] تعدادی فریم پشت سر هم به طول ۶۴ بایت بر روی کانال با نرخ خطای 10^{-7} منتقل گردد. به طور متوسط چند فریم در هر ثانیه در اثر خطا آسیب خواهد دید؟

۳۰. یک شبکه 802.16 دارای کانالی با عرض باند 20MHz است. در هر ثانیه چند بیت برای یک ایستگاه قابل ارسال خواهد بود؟

۳۱. استاندارد IEEE 802.16 از چهار رده خدمات پشتیبانی می‌کند. کدامیک از این رده‌های خدمات، بهترین انتخاب برای ارسال تصاویر ویدیوئی **فشرده‌نشده** می‌باشد؟

۳۲. دو دلیل بیاورید که چرا برخی شبکه‌ها ممکن است از کدهای تصحیح خطا به جای فرآیند کشف خطا و ارسال مجدد استفاده نمایند.

۳۳. در شکل ۴-۳۵ می‌بینیم که یک ابزار مبتنی بر بلوتوث می‌تواند به طور همزمان درون دو پیکونت قرار داشته باشد. آیا دلیلی وجود دارد که یک ابزار نتواند بطور همزمان نقش گره اصلی (Master) را در هر دو پیکونت ایفا کنند؟

۳۴. شکل ۴-۲۵، چندین پروتکل لایهٔ فیزیکی را نشان می‌دهد. کدام یک از آنها به پروتکل لایهٔ فیزیکی در بلوتوث نزدیکتر است؟ کدام یک از آنها بیشترین اختلاف را دارد؟

۳۵. بلوتوث از دو نوع لینک بین گره اصلی (Master) و گره‌های پیرو (Slave) پشتیبانی می‌کند. این دو کدام است و هر کدام برای چه کاری در نظر گرفته شده‌اند؟

۳۶. فریمهای Beacon در گونه‌ای از شبکهٔ 802.11 که مبتنی بر روش «طیف گسترده با پرش فرکانسی» (Frequency Hopping Spread Spectrum) است زمانی به نام Dwell time را مشخص می‌کند. آیا به نظر شما در فریمهای مشابه Beacon در بلوتوث نیز زمانی به نام dwell time وجود دارد؟ پاسخ خود را شرح دهید.

۳۷. به شبکه‌های LAN متصل به هم که در شکل ۴-۴۱ نشان داده شده دقت کنید. فرض کنید ماشینهای میزبان a و b بر روی LAN 1 و c برروی LAN 2 و d برروی LAN 8 واقع هستند. در ابتدا جداول Hash پلها خالی است و «درخت پوشای» نشان داده شده در شکل ۴-۴۱-ب به کار گرفته شده است. نشان دهید که جداول پلهای مختلف در اثر هر یک از رخدادهای زیر که به ترتیب حروف الفبا اتفاق می‌افتند چگونه تغییر می‌کند.

الف) a برای d ارسال می‌کند.
ب) c برای a فریمی می‌فرستد.
ج) d برای c فریمی می‌فرستد.
د) d به LAN 6 نقل مکان می‌کند.
ه) d برای a فریمی می‌فرستد.

۳۸. یکی از تبعات استفاده از «درخت پوشا» برای هدایت فریمها در یک LAN گسترش‌یافته آن است که برخی از پلها در فرآیند هدایت فریمها نقشی ایفاء نخواهند کرد. در شکل ۴-۴۴ سه تا از اینگونه پلها را مشخص کنید. آیا دلیلی دارد که چنین پلهایی را در شبکه داشته باشیم، حتی وقتی در هدایت فریمها نقشی ایفا نمی‌کنند؟

۳۹. تصور کنید که یک سوئیچ دارای چند «کارت خط» (Line Card) است و هر یک از کارتها چهار خط

ورودی دارند. فرض کنید که بطور متناوب فریمهایی که به یکی از خطوط کارت وارد می‌شوند از خط دیگری بر روی همان کارت خارج می‌گردند؛ طراح سوئیچ چه راهکاری در مواجهه با این مسئله پیش رو دارد؟

۴۰. یک سوئیچ که برای به کارگیری در اترنت سریع طراحی شده، دارای یک Backplane است که می‌تواند 10Gbps را منتقل کند. در سنگین‌ترین بار این سوئیچ قادر به انتقال چند فریم در هر ثانیه است؟

۴۱. به شبکهٔ شکل ۴-۴۹-الف دقت کنید. اگر ماشین J به ناگاه «سفید» شود [یعنی به VLAN سفید بپیوندد] آیا تغییری در برچسبها لازم است؟ اگر این چنین است چگونه؟

۴۲. به طور مختصر تفاوت بین سوئیچهای نوع «ذخیره و هدایت» (Store & Forward) و سوئیچهای Cut Through را توضیح بدهید.

۴۳. سوئیچهای نوع «ذخیره و هدایت» در مقایسه با سوئیچهای Cut Through از دیدگاه آسیب رسیدن به فریمها برتر هستند. توضیح بدهید که این برتری از کجا ناشی می‌شود؟

۴۴. برای آنکه شبکه‌های VLAN شروع به کار کنند باید جداول پلها و سوئیچها تنظیم و پیکربندی شوند. اگر در شکل ۴-۴۹-الف به جای استفاده از شبکهٔ مبتنی بر کابل چنداتصالی، در VLAN از هاب استفاده شود آیا باز هم به تنظیم این جداول نیاز است؟ آیا هابها نیز نیاز به پیکربندی جدول دارند؟ چرا بله یا چرا نه؟

۴۵. در شکل ۴-۵۰، شبکهٔ موجود و قدیمی سمت راست به یک سوئیچ سازگار با VLAN متصل شده است؛ آیا امکان دارد در اینجا نیز از یک سوئیچ قدیمی استفاده کرد؟ اگر جواب منفی است چرا؟

۴۶. برنامه‌ای بنویسید تا رفتار پروتکل CSMA/CD را در شبکه اترنت (در شرایطی که پس از ارسال یک فریم، N ایستگاه آماده ارسال هستند) شبیه‌سازی کند. برنامهٔ شما باید زمانهایی را که هر ایستگاه موفق می‌شود ارسال فریم خود را آغاز کند گزارش نماید. فرض کنید که به ازای هر برش ۵۱/۲میکروثانیه‌ای یک تیک ساعت اتفاق می‌افتد و کشف تصادم و ارسال سیگنال نویز (دنبالهٔ Jam پس از کشف تصادم) جمعاً ۵۱/۲ میکروثانیه طول می‌کشد. فریمها می‌توانند بیشترین طول مجاز را داشته باشند.

۵ لایه شبکه

وظیفهٔ لایهٔ شبکه آن است که بسته‌های داده را به هر طریق از مبداء به مقصد برساند. هر بسته برای رسیدن به مقصد ممکن است از چندین مسیریاب (Router) در میانه راه گذر کند. این عملکرد به وضوح با عملکرد لایهٔ پیوند داده (که هدف آن انتقال فریمها از انتهای یک سیم به نقطهٔ دیگر آن است) تفاوت دارد. بنابراین لایهٔ شبکه تحتانی‌ترین لایه‌ای است که با انتقال «انتها به انتها» (End-To-End) سر و کار دارد.

لایهٔ شبکه برای نیل به اهداف خود، موظف است از توپولوژی **«زیرشبکه ارتباطی»**[1] (یا به عبارتی مجموعهٔ تمام مسیریابها) اطلاع داشته باشد و با استفاده از این دانش مسیری مناسب را برگزیند. همچنین باید مراقب باشد تا از تحمیل بار اضافی بر روی برخی از خطوط ارتباطی و مسیریابها (در حالی که برخی دیگر آزاد هستند) اجتناب نماید. نهایتاً وقتی که مبداء و مقصد یک بسته در دو شبکه با پروتکلهای متفاوت و ناسازگار قرار گرفته‌اند مشکلات جدیدی بروز می‌کند که حل آنها بر عهدهٔ لایهٔ شبکه گذاشته شده است. در این فصل به مطالعه و تشریح برخی از این موارد خواهیم پرداخت. تمرکز اصلی ما بر اینترنت و لایهٔ شبکهٔ آن یعنی IP است، هر چند شبکه‌های بی‌سیم را نیز خاطرنشان خواهیم نمود.

۱ـ۵ مسائل طراحی لایهٔ شبکه

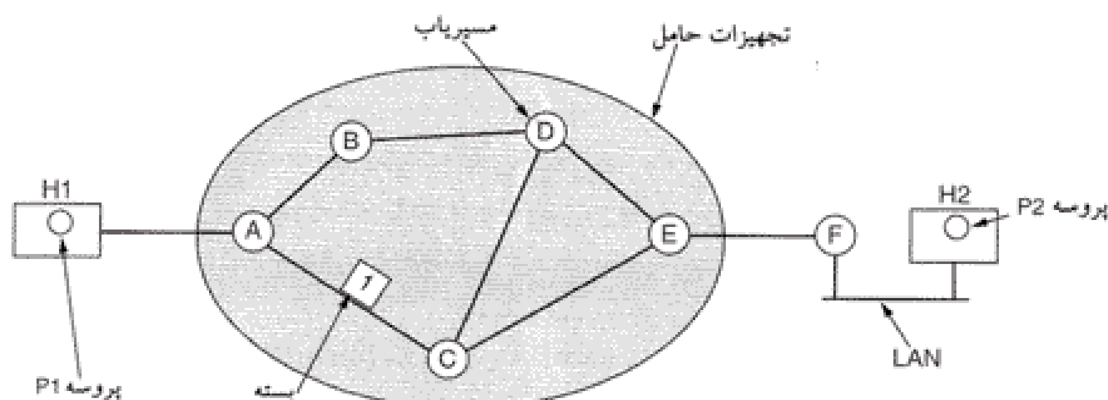
در بخشهای آتی مقدمه‌ای خواهیم داشت بر مسائلی که طراحان لایهٔ شبکه با آنها دست به گریبان خواهند بود. این مسائل شامل خدماتی است که به لایهٔ انتقال ارائه می‌گردد، یا به طراحی داخلی «زیرشبکه» (Subnet) مربوط می‌شود.

۱ـ۱ـ۵ هدایت (سوئیچینگ) بسته به روش «ذخیره و هدایت»

قبل از آنکه به تشریح جزئیات لایهٔ شبکه بپردازیم شاید مروری بر حیطه و زمینهٔ عملکرد پروتکلهای لایهٔ شبکه، خالی از لطف نباشد. الگوی کلی چنین محیطی در شکل ۵-۱ دیده می‌شود. مؤلفه‌های اصلی این سیستم عبارتند از: **تجهیزات حامل** (Carrier Equipments) که در این شکل درون بیضی سایه‌دار قرار گرفته‌اند و **تجهیزات مشتریان** (Customer's Equipments) که در خارج از بیضی قرار دارند. ماشین میزبان H1 از طریق یک خط اجاره‌ای، مستقیماً به یکی از مسیریابهای حامل یعنی A، متصل شده است. در مقابل H2 بر روی یک شبکهٔ LAN

---

۱. در این فصل به کرات از واژهٔ «زیرشبکه» به معنای Communication Subnet (مجموعهٔ تمام مسیریابهای شبکه و خطوط ارتباطی بین آنها) بهره خواهیم گرفت. ـم



شکل ۵-۱. محیطی که پروتکلهای لایهٔ شبکه در آن عمل می‌کنند.

قرار گرفته که دارای یک مسیریاب است و تحت فرمان مشتری عمل می‌کند. مسیریاب F، دارای یک خط اجاره‌ای با یکی از «تجهیزات حامل» است. ما در این شکل، مسیریاب F را خارج از بیضی در نظر گرفته‌ایم چرا که عضو بخش حامل [یعنی زیرشبکهٔ اصلی] نیست ولیکن ممکن است از لحاظ ساختار، نرم‌افزار و پروتکل تفاوتی با مسیریابهای حامل نداشته باشد. اینکه آیا چنین مسیریابی متعلق به زیرشبکه هست یا نه قابل بحث است ولیکن با اهدافی که در این فصل مدّنظر داریم مسیریابهای سمت مشتری را نیز به عنوان بخشی از زیرشبکه در نظر می‌گیریم چرا که آنها نیز همانند مسیریابهای حامل از الگوریتمهای مشابهی استفاده می‌کنند (و اهمّ کار ما نیز الگوریتمها هستند).

از این تجهیزات به نحو زیر استفاده می‌شود: یکی از ماشینهای میزبان که بسته‌ای را برای ارسال آماده دارد آن را برای نزدیکترین مسیریاب می‌فرستد، خواه این مسیریاب بر روی LAN خودش واقع باشد و خواه از طریق یک خط اجاره‌ای مستقیماً به زیرشبکهٔ حامل متصل شده باشد. بستهٔ ارسالی در آن مسیریاب موقتاً ذخیره می‌شود تا بطور کامل دریافت و کد کشف خطای آن، بررسی گردد؛ سپس به مسیریاب بعدی واقع بر مسیر، هدایت می‌شود و این روند آنقدر تکرار می‌شود تا بسته به ماشین مقصد رسیده و تحویل داده شود. این مکانیزم همان روش **«ذخیره و هدایت»** است که در فصول قبلی بدانها پرداختیم.

۵-۱-۲ خدمات ارائه شده برای لایهٔ انتقال

لایهٔ شبکه خدماتی را برای لایهٔ انتقال تدارک می‌بیند که توسط واسط میانی لایهٔ شبکه و انتقال[۱] عرضه می‌شود. سؤال مهم آن است که لایهٔ شبکه چه نوع خدماتی را برای لایهٔ انتقال تدارک می‌بیند؟ خدمات لایهٔ شبکه با در نظر داشتن اهداف زیر طراحی شده‌اند:

۱. این خدمات بایستی مستقل از تکنولوژی بکار رفته در مسیریاب باشد.
۲. لایهٔ انتقال باید از درگیری با جزئیاتی مثل تعداد، نوع و توپولوژی مسیریابهای موجود دور نگه داشته شود.
۳. آدرسهای شبکه که در اختیار لایهٔ انتقال قرار می‌گیرند بایستی از ساختار متحدالشکل و استاندارد برخوردار باشند (خواه در شبکهٔ LAN و خواه در شبکهٔ WAN).

با در نظر داشتن این اهداف، طراحان لایهٔ شبکه در تدوین شرح مبسوط خدماتی که باید به لایهٔ انتقال عرضه شود آزادی عمل دارند. این آزادی عمل اغلب به مناقشات دیرینهٔ دو گروه با طرز فکر مخالف دامن می‌زند. مناقشه

۱. Network Layer/Transport Layer Interface

بر سر آن است که آیا لایهٔ شبکه باید خدمات اتصال‌گرا (Connection Oriented) ارائه کند یا خدمات بدون اتصال (Connectionless).

یکی از طرفین (به رهبری دست‌اندرکاران اینترنت) استدلال می‌کند که وظیفهٔ یک مسیریاب هدایت بسته‌هاست و نه چیز دیگر! از دیدگاه آنها (به پشتوانهٔ سی سال تجربهٔ واقعی با یک شبکه کامپیوتری حقیقی و در حال کار) زیرشبکه ذاتاً غیرقابل اعتماد است و چگونگی طراحی آن اهمیت ندارد. بدین ترتیب ماشینهای میزبان باید این حقیقت را بپذیرند که زیرساخت ارتباطی شبکهٔ آنها غیرقابل اعتماد است و خودشان موظفند عملیات نظارت بر خطاهای احتمالی (شامل کشف و تصحیح) و همچنین کنترل جریان (Flow Control) را بر عهده بگیرند.

این دیدگاه سریعاً بدانجا منتهی می‌شود که خدمات لایهٔ شبکه باید **«بدون اتصال»** بوده و فقط عملیات پایه و ابتدایی SEND PACKET و RECEIVE PACKET و تعداد کمی دیگر از همین عملیات را انجام بدهند. خصوصاً نباید عملیات مرتب‌سازی بسته‌ها [از لحاظ ترتیب ارسال] و کنترل جریان انجام شود چراکه ماشینهای میزبان در هر حال چنین کاری را انجام می‌دهند و انجام دوبارهٔ یک کار فایدهٔ چندانی نخواهد داشت. همچنین هر بسته باید مشخصات کامل آدرس مقصد را با خود داشته باشد چراکه بسته‌های ارسالی باید مستقل از یکدیگر و فارغ از آنکه مسیریاب قبلی کدام بوده، هدایت شوند.

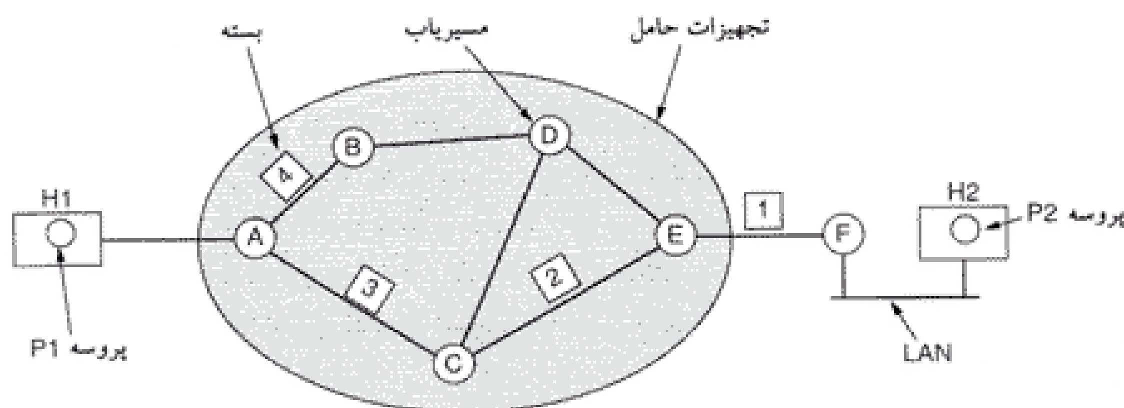
طرف مقابل این مناقشه (به رهبری شرکتهای مخابراتی) دلیل می‌آورد که زیرشبکه باید خدماتی اتصال‌گرا و قابل اعتماد ارائه کند. اینان نیز مدعیند که سابقهٔ درخشان و صد سالهٔ آنها در سیستمهای جهانی تلفن، دلیل عالی و رهنمای راه آنهاست. از دیدگاه این گروه، کیفیت خدمات (QoS) عمده‌ترین مسئله است و اگر زیرشبکه اتصال‌گرا نباشد دستیابی به کیفیت خدمات (بالاخص برای ترافیک داده‌های بی‌درنگ مثل صدا و تصویر) دشوار خواهد بود.

این دو طرف دعوا نمونه‌های خاص خود یعنی اینترنت و ATM را در دست داشتند: اینترنت خدمات لایهٔ شبکه را بدون اتصال عرضه می‌کند در حالی که ATM در لایهٔ شبکه خدماتی اتصال‌گرا ارائه می‌نماید. ولیکن اشاره به این نکته جالب است که هر چه تضمین کیفیت خدمات بیشتر و بیشتر اهمیت پیدا می‌کند، اینترنت نیز رشد می‌نماید. خصوصاً آنکه در راه کسب ویژگیهایی است که عموماً منتسب به خدمات اتصال‌گرا هستند؛ بعداً در این خصوص اشاراتی خواهیم داشت. در خلال مطالعهٔ شبکه‌های VLAN در فصل چهارم نمونه‌ای از این رشد و تکامل را ارائه کردیم.

۵-۱-۳ پیاده‌سازی خدمات بی‌اتصال

پس از نگاهی بر دو رده از خدماتی که لایهٔ شبکه می‌تواند به کاربران عرضه نماید وقت آن فرا رسیده که ببینیم در درون این لایه چه می‌گذرد. بسته به نوع خدمات ارائه شده، دو نوع معماری متفاوت ممکن خواهد بود. اگر خدمات بدون اتصال عرضه شود بسته‌ها بطور مجزا و مستقل به درون زیرشبکه تزریق و مستقل از یکدیگر هدایت و مسیریابی می‌شوند و هیچگونه تنظیمات قبلی نیاز نیست. در چنین حالتی عموماً به بسته‌ها «دیتاگرام» (Datagram) و به زیرشبکه نیز **«زیرشبکه دیتاگرام»** اطلاق می‌شود. اگر خدمات اتصال‌گرا عرضه شود قبل از ارسال هر گونه بستهٔ داده بایستی از قبل یک مسیر بین مسیریاب مبداء و مسیریاب مقصد تنظیم و ایجاد شود. چنین ارتباطی اصطلاحاً VC (مخفف Virtual Circuit) نامیده می‌شود (مترادف با ایجاد یک مدار فیزیکی در شبکهٔ تلفن) و به زیرشبکه نیز، **«زیرشبکهٔ مدار مجازی»** (Virtual Circuit Subnet) گفته می‌شود. در این بخش ابتدا به زیرشبکه‌های دیتاگرام می‌پردازیم و در بخش بعدی زیرشبکه‌های مدار مجازی را مطالعه خواهیم نمود.

حال بیائید به بررسی عملکرد زیرشبکه مبتنی بر دیتاگرام بپردازیم. فرض کنید پروسهٔ P1 در شکل ۵-۲ پیامی طولانی برای پروسهٔ P2 دارد. او این پیام را به لایهٔ انتقال سپرده و از او می‌خواهد که پیام را به پروسهٔ P2 بر روی



جدول A (initially)		جدول A (later)		جدول C		جدول E	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

(مقصد) Dest. Line (خط)

شکل ۵-۲. مسیریابی در یک زیرشبکهٔ دیتاگرام.

ماشین میزبان H2 تحویل بدهد. کد نرم‌افزار لایهٔ انتقال که عموماً بخشی از سیستم عامل به حساب می‌آید اجرا شده و سرآیند لایهٔ انتقال در ابتدای پیام درج و نتیجه ، تحویل لایهٔ شبکه می‌شود که آن نیز یک پروسهٔ اجرایی (روال اجرایی) در سیستم عامل است.

فرض کنیم که پیام چهار برابر طولانی‌تر از اندازهٔ مجاز بسته‌های داده باشد؛ لذا لایهٔ شبکه مجبور است آنها را به چهار بستهٔ ۱ و ۲ و ۳ و ۴ بشکند و هر یک از آنها را بطور مجزا و از طریق یک پروتکل نقطه‌به‌نقطه (نظیر PPP)[۱] برای مسیریاب A می‌فرستد. از اینجا به بعد ادامهٔ کار بر عهدهٔ زیرشبکهٔ حامل قرار می‌گیرد. هر مسیریاب دارای جدولی داخلی است که مشخص می‌کند برای رسیدن به تمام مسیریابهای مقصد در شبکه، بسته باید بر روی کدام خط خروجی [کدام مسیریاب بعدی] ارسال شود. هر یک از «درایه‌های» این جدول (Table Entry) شامل یک جفت آیتم اطلاعاتی است که یکی «مقصد» و دیگری خط خروجی برای رسیدن بدان مقصد را تعیین می‌نماید. برای هدایت بسته فقط می‌توان از خطوطی که مستقیماً به یک مسیریاب مجاور متصلند بهره گرفت. به عنوان مثال در شکل ۵-۲ ، مسیریاب A فقط دو خط خروجی (به مسیریابهای B و C) دارد و طبعاً هر یک از بسته‌های ورودی باید برای یکی از این دو مسیریاب ارسال گردد، حتی اگر مقصد نهایی به مسیریابی کاملاً متفاوت متصل باشد. جدول مسیریابی ابتدایی A با عنوان "Initialy" ، در شکل ۵-۲ مشخص شده است.

پس از ورود بسته‌های ۱ و ۲ و ۳ به مسیریاب A ، موقتاً در حافظه ذخیره می‌شوند (تا کدهای کشف خطای آنها بررسی شود). سپس براساس جدول مسیریابی A ، این بسته‌ها به سمت مسیریاب C هدایت می‌شوند. به همین ترتیب و در ادامهٔ طی مسیر، بستهٔ ۱ از E و نهایتاً F می‌گذرد. پس از آنکه بسته به F رسید درون فریم لایهٔ پیوند داده

۱. پروتکل PPP یک پروتکل برای خطوط نقطه‌به‌نقطه است که در لایهٔ پیوند داده‌ها عمل می‌کند یعنی یک فریم را بر روی کانالی واحد و غیر مشترک، از نقطه‌ای به نقطهٔ دیگر تحویل می‌دهد. به فصل سوم مراجعه کنید. -م.

جاسازی (کپسوله) شده و از طریق شبکهٔ LAN برای H2 ارسال می‌گردد. بسته‌های ۲ و ۳ نیز به همین طریق مسیریابی و هدایت می‌شوند.

ولیکن (به عنوان مثال) برای بستهٔ ۴ اتفاق متفاوتی می‌افتد. وقتی این بسته به A می‌رسد اگرچه به مقصد F طی مسیر می‌کند ولیکن برخلاف قبل به سوی مسیریاب B ارسال می‌شود. به دلایل متعدد A تصمیم گرفته تا بستهٔ ۴ را از مسیری متفاوت با مسیر سه بستهٔ قبلی، به سمت F بفرستد. شاید این مسیریاب متوجه ازدیاد ترافیک و ازدحام بر روی مسیر ACE شده و جدول مسیریابی خود را به جدولی جدید (که در شکل با عنوان Later مشخص شده) اصلاح و بهنگام‌سازی کرده است. الگوریتمی که مدیریت این جداول را بر عهده دارد و در خصوص مسیریابی تصمیم می‌گیرد اصطلاحاً «**الگوریتم مسیریابی**» (Routing Algorithm) نامیده می‌شود. الگوریتمهای مسیریابی یکی از اصلیترین مفاد این فصل هستند.

۵-۱-۴ پیاده‌سازی خدمات اتصال‌گرا

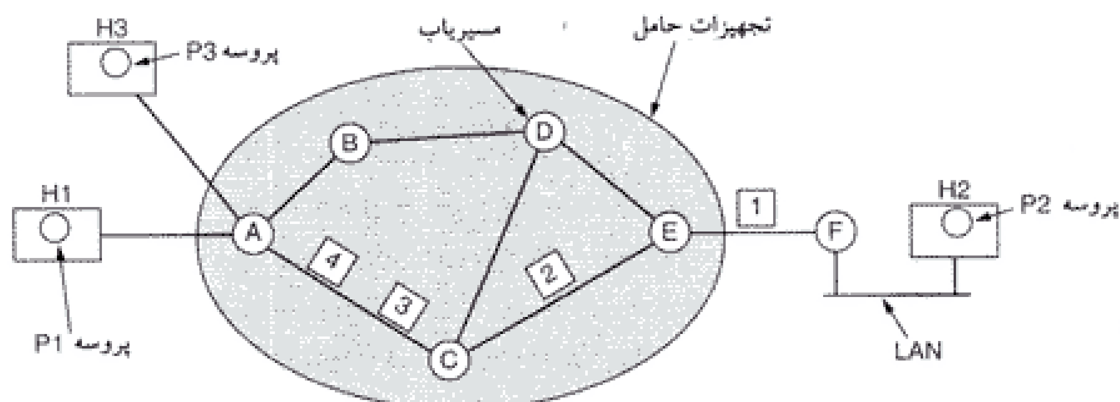
برای عرضهٔ خدمات اتصال‌گرا، به زیرشبکهای مبتنی بر «مدار مجازی» (Virtual Circuit) نیازمندیم. حال ببینیم که چگونه کار می‌کند: ایدهٔ اصلی در مدار مجازی آن است که برخلاف مثال شکل ۵-۲ ، از انتخاب مسیرهای جدید برای هر بسته اجتناب شود. در عوض وقتی یک «**اتصال**» ایجاد شد، به عنوان بخشی از عملیات تنظیم اتصال (Connection Setup) ، مسیری بین ماشین مبداء و ماشین مقصد انتخاب می‌شود و مشخصات آن در جدول داخلی هر مسیریاب درج می‌گردد. دقیقاً مشابه با سیستم تلفن پس از برقراری اتصال، تمام بسته‌ها از طریق همین مسیر هدایت خواهند شد. پس از آنکه اتصال برقرارشده خاتمه یافت مدار مجازی متناظر نیز پایان یافته و حذف می‌شود. در خدمات اتصال‌گرا هر بسته با خود یک شمارهٔ شناسایی حمل می‌کند که مشخص می‌کند به کدام مدار مجازی تعلق دارد.[1]

به عنوان مثال، به وضعیت نشان داده شده در شکل ۵-۳ توجه کنید. در اینجا ماشین میزبان H1 اتصال شماره ۱ را با ماشین میزبان H2 برقرار نموده است. مشخصهٔ این اتصال به عنوان اولین درایه (Entry) در جداول مسیریابی درج شده است. اولین سطر از جدول A بیان می‌کند که اگر بسته‌ای با شمارهٔ شناسایی ۱ از H1 دریافت شود باید با همین شمارهٔ شناسایی برای مسیریاب C ارسال گردد. به روش مشابه مسیریاب C مطابق با اولین درایهٔ خود، بسته را با شمارهٔ شناسایی ۱ به سوی مسیریاب E هدایت می‌نماید.

حال بررسی کنیم که اگر H3 نیز بخواهد اتصالی را با H2 برقرار کند چه اتفاقی می‌افتد. او نیز شماره شناسایی اتصال را ۱ در نظر می‌گیرد (چرا که این اتصال، اولین اتصالی است که او برقرار کرده است) و سپس از زیرشبکه می‌خواهد که برایش مداری مجازی ایجاد کند. انجام این تقاضا منجر به اضافه شدن سطر دوم به جداول است. دقت کنید که در اینجا یک تناقض وجود دارد و آن هم اینکه اگرچه A می‌تواند بسته‌هایی که با شمارهٔ شناسایی ۱ از H1 می‌رسند را از بسته‌هایی با همین شماره که از H3 می‌رسند تمیز بدهد ولی مسیریاب C قادر به چنین کاری نیست [چرا که هر دو با یک شماره و از A می‌رسند]. به همین دلیل A برای ترافیک خروجی که در قالب اتصال ۲ خارج می‌شوند شمارهٔ شناسایی متفاوتی را در نظر می‌گیرد. برای اجتناب از تناقضاتی از این دست، مسیریابها باید بتوانند شمارهٔ شناسایی اتصال هر بستهٔ خروجی را تغییر بدهند.

---

۱. اتصال را در ذهن خود با یک تماس تلفنی قیاس کنید. ابتدا شماره می‌گیرید، تماس شما برقرار می‌شود، تا زمان دلخواه گفتگو می‌کنید و نهایتاً تماس را قطع می‌نمایید. اتصال در زیرشبکهٔ مدار مجازی بمعنای هماهنگی قبلی بین مبداء، مقصد و مسیریابهای میانی تلقی می‌شود. به مسیری که با هماهنگی قبلی ایجاد می‌شود یک «مدار مجازی» می‌گویند و دارای یک شماره شناسایی است. به این شمارهٔ شناسایی اصطلاحاً «شناسهٔ مدار مجازی» یا «شناسهٔ اتصال» (Connection ID) اطلاق می‌شود. -م



جدول A

In		Out	
H1	1	C	1
H3	1	C	2

جدول C

A	1	E	1
A	2	E	2

جدول E

C	1	F	1
C	2	F	2

شکل ۵-۳. مسیریابی در یک زیرشبکهٔ مدار مجازی.

۵-۱-۵ مقایسهٔ زیرشبکه‌های مدار مجازی و دیتاگرام

هر یک از روشهای مدار مجازی و دیتاگرام حامیان و منتقدین خود را دارند. حال تلاش می‌کنیم مباحثات آنها را به طور خلاصه ارائه نماییم. محورهای اصلی این مباحثات در شکل ۵-۴ فهرست شده‌اند؛ هر چند ممکن است سفسطه کنندگان بتوانند برای هر یک از موارد این جدول، مثال نقضی پیدا کنند.

در هر زیرشبکه می‌توان اصولی را برای قیاس زیرشبکه‌های مدار مجازی و دیتاگرام مطرح کرد. یکی از آنها مسئله میزان حافظه مسیریاب در مقایسه با پهنای باند تلفاتی است. روش مدار مجازی اجازه می‌دهد که بسته‌ها به جای همراه داشتن آدرسهای کامل فقط شمارهٔ مدار مجازی را با خود داشته باشند. هر گاه بسته‌ها نسبتاً کوتاه باشند وجود آدرس کامل در هر بسته ممکن است حجم سربار زیادی را تحمیل کرده و بدین ترتیب بخشی از پهنای باند مفید هدر می‌رود. در عوض، هزینه‌ای که برای بکارگیری مدار مجازی پرداخت می‌شود فضای حافظه‌ای است که جدول مشخصات مدارات مجازی در درون مسیریاب، به خود اختصاص می‌دهد. بسته به قیمت پهنای باند کانالهای مخابراتی در مقایسه با حافظهٔ مسیریاب یکی از این دو ارزانتر تمام می‌شود.

یکی دیگر از مسائل قیاسی، «زمان تنظیم و ایجاد مدار مجازی» در مقایسه با زمان جستجو و تحلیل آدرسها است.[۱] استفاده از مدار مجازی منوط به طی مراحل تنظیم مسیر است که فرآیندی زمان‌بر بوده و منابع زیرشبکه را مصرف می‌کند. با این وجود در زیرشبکهٔ مبتنی بر مدار مجازی، تشخیص آنکه چه کاری با یک بستهٔ داده باید انجام شود ساده است: مسیریاب از شمارهٔ شناسایی مدار مجازی به عنوان اندیس جدول مسیریابی استفاده می‌کند تا مسیری که بسته باید طی کند مشخص شود. در زیرشبکهٔ دیتاگرام برای جستجو و پیدا کردن درایهٔ متناظر با مقصد، به روالی پیچیده‌تر نیاز است.

مورد دیگر، فضای مورد نیاز جدول مسیریابی در حافظهٔ مسیریاب است. یک زیرشبکه دیتاگرام نیازمند آن است که به ازای هر مقصد در شبکه، یک درایه (Entry) در جدول مسیریابی خود داشته باشد، در حالی که در

۱. Setup Time versus Address Parsing Time

مورد	زیرشبکه دیتاگرام	زیرشبکه مدارمجازی
تنظیم مدار (Circuit Setup)	نیازی نیست	نیاز است.
آدرس‌دهی	هر بسته آدرس دقیق و کامل مبداء و مقصد را با خود حمل می‌کند.	هر بسته فقط یک شماره کوتاه مدار مجازی (VC) را با خود دارد.
اطلاعات وضعیت	مسیریاب نیازی به نگهداری اطلاعاتی در خصوص وضعیت هر اتصال ندارد.	به ازای هر مدار مجازی تمام مسیریابها باید اطلاعاتی در خصوص وضعیت آن نگاه دارند.
مسیریابی	هر بسته بطور مستقل مسیریابی می‌شود.	مسیر فقط یکبار و آنهم در هنگام تنظیم مدار مجازی انتخاب می‌شود و تمام بسته‌ها از همان مسیر حرکت می‌کنند.
تاثیر خرابی مسیریاب	بی‌تاثیر، مگر در مورد بسته‌هایی که در حین خرابی از بین رفته‌اند.	تمام مدارات مجازی که از مسیریاب خراب شده می‌گذشته‌اند قطع می‌شوند.
تضمین کیفیت خدمات	دشوار	اگر برای هر مدار مجازی منابع لازم از قبل تخصیص یابد، بسیار آسان است.
کنترل ازدحام	دشوار	اگر برای هر مدار مجازی منابع لازم از قبل تخصیص یابد، بسیار آسان است.

شکل ۵-۴. مقایسهٔ زیرشبکه‌های دیتاگرام و مدار مجازی.

زیرشبکهٔ مدار مجازی، فقط به ازای هر اتصال به یک درایه نیاز است. ولیکن این حُسن تا حدودی گمراه‌کننده و توخالی است چرا که بسته‌های تنظیم اتصال [که برای اولین بار ارسال می‌شوند] بهر نحو باید همانند روش دیتاگرام یکبار مسیریابی شوند و آنها نیز دارای آدرسهای کامل مقصد هستند. [یعنی در زیرشبکهٔ مدار مجازی هم به جدول مسیریابی و هم به جدول مدارات مجازی نیاز است. -م]

مدارهای مجازی دارای محاسنی در خصوص تضمین «کیفیت خدمات» (QoS) هستند و از بروز ازدحام (congestion) در زیرشبکه جلوگیری می‌کنند چرا که «منابع» (شامل بافر، پهنای باند و سیکلهای CPU) می‌تواند پیشاپیش و در هنگام ایجاد اتصال، رزرو شود. بعداً، به محض دریافت بسته‌ها، پهنای باند مورد نیاز و ظرفیت لازم در مسیریاب، مهیا و آماده است. در زیرشبکه‌های دیتاگرام، اجتناب از ازدحام دشوارتر است.

در «سیستمهای پردازش تراکنشی»[1] ، (همانند وقتی که خرید با کارتهای اعتباری بررسی می‌شوند) مسئله تنظیم یک مدار مجازی و سپس حذف آن ممکن است کاربرد این روش را از اعتبار ساقط کند. [به عبارت بهتر در سیستمهایی که تعامل و مبادلهٔ داده با ماشینها، کوتاه و سریع است ایجاد یک مدار مجازی و ختم آن سربار زیادی به شبکه تحمیل می‌نماید. -م] اگر بخش اعظم ترافیک از این نوع باشد استفاده از مدار مجازی درون یک زیرشبکه چندان مناسب نیست. در طرف مقابل هرگاه حجم داده‌های ارسالی انبوه باشد، مدارهای مجازی دائم که به صورت دستی تنظیم می‌شوند و برای ماهها یا سالها تغییر نمی‌کنند می‌تواند سودمندتر باشد.

مدارهای مجازی مشکل «آسیب‌پذیری» دارند. اگر یک مسیریاب از کار بیفتد و حافظه خود را از دست بدهد (حتی در صورتی که چند ثانیه بعد به زیرشبکه برگردد) ، قطعاً تمام مدارهای مجازی که از آن عبور می‌کنند، از دست خواهند رفت. در مقابل اگر یک مسیریاب مبتنی بر دیتاگرام از کار بیفتد تنها کاربرانی آسیب می‌بینند که

1. Transaction Processing System

بسته‌های آنها هنگام بروز مشکل، درون حافظهٔ مسیریاب در صف منتظر بوده و بر حسب آنکه آیا دریافت بسته‌های داده قبلاً اعلام شده باشد یا نه، حتی ممکن است فقط بخشی از آنها با مشکل مواجه شوند. از دست رفتن یک خط مخابراتی برای مدارات مجازی که از آن خط استفاده کرده‌اند بحران‌زاست در حالی که اگر از روش دیتاگرام استفاده شده باشد این اشکال براحتی جبران خواهند شد. در روش دیتاگرام مسیریابها مجازند ترافیک را بر روی زیرشبکه توزیع و موازنه کنند لذا ممکن است در حین ارسال یک دنبالهٔ طولانی از بسته‌ها مسیرها عوض شوند.

۲-۵ الگوریتمهای مسیریابی

وظیفهٔ اصلی لایهٔ شبکه، مسیریابی و هدایت بسته‌های داده از ماشین مبداء به ماشین مقصد است. در اکثر زیرشبکه‌ها، بسته‌ها برای طی مسیر خود بایستی چندین **«گام»** (HOP) راه بپیمایند.[1] الگوریتمی که این مسیرها را انتخاب می‌کنند و همچنین ساختمان دادهٔ مورد استفاده، یکی از زمینه‌های مهم در طراحی لایهٔ شبکه محسوب می‌شود.

«الگوریتمهای مسیریابی» آن بخش از نرم‌افزار لایهٔ شبکه‌اند که مسئولیت دارند در خصوص خط خروجی که یک بسته ورودی باید بر روی آن ارسال شود، تصمیم‌گیری کنند. اگر در داخل زیرشبکه از روش دیتاگرام استفاده شده باشد این تصمیم‌گیری باید به ازای هر بستهٔ دریافتی از نو تکرار شود چراکه در هر لحظه ممکن است بهترین مسیر تغییر نماید. اگر زیرشبکه از روش مدار مجازی بهره گرفته باشد، تصمیم‌گیری در خصوص مسیرها فقط یکبار و آنهم در هنگام تنظیم و ایجاد هر مدار مجازی جدید صورت می‌گیرد و طبعاً بسته‌های داده، همگی از مسیری که قبلاً ایجاد شده هدایت می‌شوند. این حالت گاهی روش **«مسیریابی مبتنی بر نشست»** (Session Routing) نامیده می‌شود چرا که مسیر ایجاد شده در خلال نشست کاربر برقرار خواهد ماند؛ (مثلاً در حین نشست از راه دور با ترمینال ـLogin sessionـ یا نشست انتقال فایل).

گاهی تمایز قائل شدن بین فرآیند «مسیریابی» (که به تصمیم‌گیری در خصوص مسیرهای بهینه اطلاق می‌شود) و فرآیند «هدایت» (Forwarding) (آنچه که با ورود بسته اتفاق می‌افتد) مفید است: می‌توانید بدینگونه بیندیشید که هر مسیریاب دارای دو پروسه در درون خود است: یکی از آنها بسته‌ها را به محض ورود پردازش کرده و از طریق جدول مسیریابی، یک خط خروجی مناسب برای آنها انتخاب می‌نماید. این پروسه همان «عمل هدایت» است. پروسهٔ دیگر موظف به پر کردن و بهنگام‌سازی محتویات جدول مسیریابی است. این همان جایی که «الگوریتمهای مسیریابی» (Routing Algorithm) به میدان می‌آیند.

فارغ از آنکه آیا برای هر بسته، مسیرها بطور مستقل و جداگانه انتخاب می‌شوند یا آنکه فقط یکبار و آن هم در حین ایجاد یک اتصال [مدار مجازی] تعیین می‌گردد، از یک الگوریتم مسیریابی ویژگیهای خاصّی انتظار می‌رود: (۱) **صحّت عملکرد** (correctness) (۲) **سادگی** (simplicity) (۳) **قابلیت تحمل** (Robustness) (۴) **پایداری** (stability) (۵) **مساوات** (Fairness) (۶) **بهینگی** (Optimality).

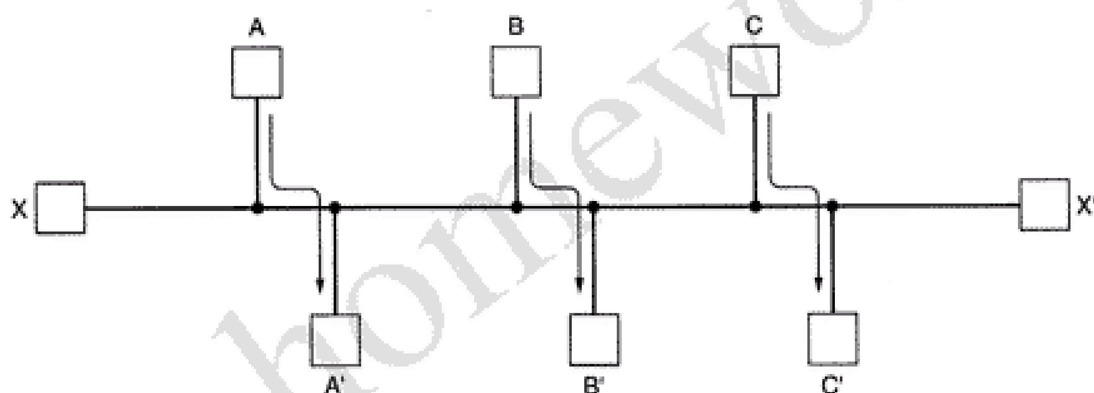
ویژگی «صحت» و «سادگی» نیازی به توضیح ندارد در حالی که نیاز به «قابلیت تحمل» در بدو امر چندان روشن به نظر نمی‌رسد. با ورود یک شبکهٔ عظیم به صحنهٔ عمل، انتظار آنست که برای سالها بدون بروز هیچگونه خرابی سیستم در سطح وسیع، به کار خود ادامه بدهد. در خلال این دورهٔ زمانی ممکن است انواع سخت‌افزارها و نرم‌افزارها از کار بیفتند. ماشینهای میزبان، مسیریابها و خطوط ارتباطی به کرّات از کار می‌افتند و توپولوژی شبکه

---

۱. به گذر بسته از یک مسیریاب، گام یا Hop گفته می‌شود

چندین بار تغییر می‌کند. الگوریتم مسیریابی باید بتواند هر گونه تغییر در توپولوژی و ترافیک را بدون ایجاد وقفه در کار ماشینهای میزبان یا نیاز به راه‌اندازی مجدد شبکه (در صورت از کار افتادن برخی از مسیریابها)، اصلاح و مدیریت نماید.

یکی دیگر از اهداف و ویژگیهای الگوریتم مسیریابی «پایداری» است. برخی از الگوریتمهای مسیریابی هرگز به یک عملکرد ثابت و پایدار همگرا نمی‌شوند بدون آنکه مدت زمان در حال کار و اجرا بودن آنها اهمیت و تأثیری داشته باشد. ویژگیهای «مساوات» و «بهینگی» ممکن است واضح به نظر برسند و مطمئناً هیچ عقل سلیمی با آن مخالف نیست ولی گاهی در مرحلهٔ عمل با یکدیگر در تناقض و تضاد هستند. به عنوان یک مثال از این تناقض، به شکل ۵-۵ نگاه کنید. فرض نمائید که ترافیک بین A و 'A، بین B و 'B و بین C و 'C بقدری است که لینک ارتباط افقی اشباع می‌شود. برای آنکه جریان کلی اطلاعات به حداکثر برسد باید ترافیک بین X و 'X قطع گردد؛ [تا بهینگی رعایت شده باشد] متأسفانه X و 'X ممکن است متوجه چنین موضوعی نباشند ولیکن باید بین «کارآیی» و «مساوات» حالتی بینابین و متعادل مدّنظر قرار بگیرد. اگر ترافیک بین X و 'X قطع شود بهینگی بدست می‌آید ولی در عوض مساوات رعایت نخواهد شد!



شکل ۵-۵. تضاد بین «بهینگی» و «مساوات».

قبل از آنکه بتوان بین «مساوات» و «بهینگی» تعادل ایجاد کرد باید ابتدا تصمیم بگیریم که در پی بهینه کردن چه چیزی هستیم! به حداقل رساندن متوسط تأخیر بسته‌ها نامزد واضحی برای بهینه شدن است ولی به حداکثر رساندن بازده مفید شبکه (Throughput) نیز نامزد دیگری است. به علاوه این دو هدف با یکدیگر در تناقض هستند چرا که مثلاً هر گاه یک سیستم صف‌بندی [همانند صفی که در هر مسیریاب ایجاد می‌شود] در ظرفیت حداکثر خود کار کند تأخیری طولانی را تحمیل خواهد کرد.[1] به عنوان یک حالت بینابین، در برخی از شبکه‌ها سعی شده تا تعداد «گامهای مسیر» (که یک بسته باید طی کند) به حداقل برسد زیرا کاهش تعداد گامها باعث کاهش تأخیر و پهنای باند مصرفی خواهد شد که نهایتاً به بهبود بازده مفید شبکه می‌انجامد.

الگوریتمهای مسیریابی را می‌توان به دو ردهٔ کلی تقسیم‌بندی کرد: «غیروفقی» (nonadaptive) و «وفقی» (adaptive). روشهای غیروفقی تصمیم‌گیری خود در خصوص مسیریابی را بر مبنای اندازه‌گیری و تخمین هوشمند ترافیک و توپولوژی فعلی شبکه، قرار نداده‌اند. در عوض برای انتخاب مسیری بین I و J (برای هر I و J مفروض)، یک مسیر از قبل و به صورت offline محاسبه شده و در هنگام راه‌اندازی شبکه درون مسیریابها بارگذاری می‌شود. به این روال اغلب «مسیریابی ایستا» (Static Routing) گفته می‌شود.

---

۱. در صف نگاه داشتن بسته‌ها می‌تواند به بازده مفید شبکه بیفزاید ولی تأخیر را نیز افزایش خواهد داد. رجوع کنید به نظریهٔ صف -م

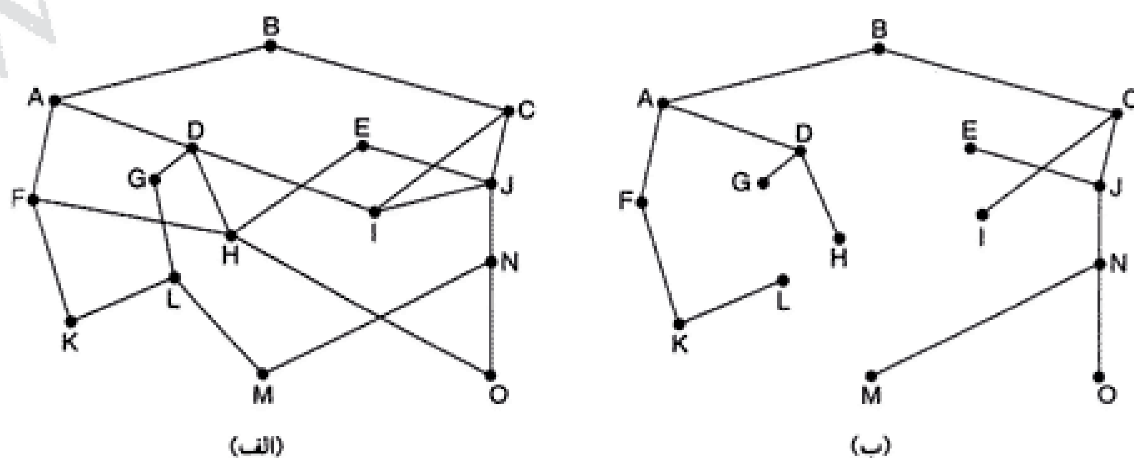
در مقابل، «الگوریتمهای وفقی» تصمیم‌گیری در خصوص مسیریابی را برحسب تغییرات توپولوژی و عموماً ترافیک لحظه‌ای، بصورت هوشمند انجام می‌دهند. الگوریتمهای مسیریابی وفقی در نحوهٔ گردآوری «اطلاعات مسیر» با یکدیگر متفاوتند (مثلاً اینکه آیا این اطلاعات به صورت محلی گردآوری می‌شود یا از مسیریابها مجاور بدست می‌آید یا از همهٔ مسیریابها می‌رسد). الگوریتمهای وفقی همچنین در پارامترهایی نظیر: (۱) زمان بهنگام‌سازی اطلاعات مسیر (مثلاً بطور متناوب هر ΔT ثانیه بهنگام شوند یا فقط زمانیکه توپولوژی عوض شود) (۲) معیاری که برای بهینه‌سازی مورد استفاده قرار می‌گیرد (مثل فاصله، تعداد گام (HOP) یا زمان تخمینی انتقال) با یکدیگر متفاوت هستند. در بخش بعدی گونه‌های مختلفی از الگوریتمهای مسیریابی اعم از پویا و ایستا را تشریح خواهیم کرد.

۵-۲-۱ اصل بهینگی

قبل از آنکه بر روی یک الگوریتم خاص متمرکز شویم، شاید پرداختن به این نکته مفید باشد که هر کسی می‌تواند بدون توجه به توپولوژی یا ترافیک شبکه یک ارزیابی و تعبیر کلی در خصوص مسیرهای بهینه ارائه بدهد. این تعبیر به نام «**اصل بهینگی**» مشهور است و بیان می‌کند که اگر مسیریاب J بر روی مسیر بهینهٔ بین مسیریاب I تا مسیریاب K واقع شده باشد بنابراین مسیر بهینه از J تا K نیز بر روی همان مسیر خواهد بود. برای بررسی این موضوع، آن قسمت از مسیر که بین I تا J قرار دارد را r_1 و مابقی مسیر را r_2 بنامید. اگر مسیری بهتر از r_2 بین J تا K وجود داشته باشد می‌توان آن را به r_1 افزود تا مسیر بهتری از I تا K ایجاد شود؛ این موضوع با اصل قضیه که بیان می‌کرد مسیر r_1r_2 بهینه است تناقض دارد.

به عنوان یک نتیجهٔ مستقیم از اصل بهینگی می‌توان اثبات کرد که مجموعهٔ مسیرهای بهینه بین تمام گره‌های مبداء و یک گره مقصد خاص، درختی را تشکیل می‌دهد که ریشهٔ آن بر روی گره مقصد قرار دارد. به چنین درختی اصطلاحاً Sink Tree گفته می‌شود که شمای آن در شکل ۵-۶ به تصویر کشیده شده است. در این شکل معیار فاصله «تعداد گام» در نظر گرفته شده است. دقت کنید که درخت Sink Tree الزاماً یکتا و منحصر به فرد نیست و ممکن است درختی متفاوت با طول مسیرهای مشابه پیدا شود. هدف اصلی الگوریتمهای مسیریابی آن است که برای تمام مسیریابها درخت Sink Tree محاسبه شده و مورد استفاده قرار بگیرد.

بدیهی است که Sink Tree، حداقل یک درخت است، بنابراین بهیچوجه دارای حلقه نیست و بدین ترتیب هر بسته پس از طی چندگام محدود و معین، تحویل مقصد خواهد شد. ولیکن در دنیای عمل همیشه کار بدین سادگی



شکل ۵-۶. (الف) یک زیرشبکه (ب) درخت Sink Tree برای مسیریاب B.

نخواهد بود. لینکها و مسیریابها می‌توانند به ناگاه از کار بیفتند و سپس مجدداً عملیات طبیعی خود را از سر بگیرند لذا مسیریابهای مختلف ممکن است توپولوژی شبکه را به گونهٔ متفاوتی ببینند.[۱] همچنین ممکن است بدین نکتهٔ ظریف بیندیشیم که آیا هر مسیریاب مجبور است مستقلاً اطلاعات لازم در خصوص توپولوژی شبکه را جمع‌آوری کرده و Sink Tree مربوطه را خودش محاسبه نماید یا آنکه این اطلاعات به روشهای دیگری بدست می‌آیند. در بخشهای آتی مختصراً بدین موضوعات خواهیم پرداخت. بهر حال، «اصل بهینگی» و «Sink Tree» روشی برای محک زدن و ارزیابی الگوریتمهای مسیریابی دیگر هستند.

۵-۲-۲ مسیریابی مبتنی بر کوتاهترین مسیر

اجازه بدهید مطالعات خود پیرامون الگوریتم‌های مسیریابی را با پرداختن به روشی آغاز کنیم که بطور گسترده‌ای از آن استفاده می‌شود چراکه ساده و فهم جزئیات آن راحت است. ایدهٔ اصلی آن است که گراف زیرشبکه را بگونه‌ای تشکیل بدهیم که در آن هر گره از گراف یک مسیریاب را مشخص می‌کند و هر یک از «کمانها» (Arcs) مشخص‌کنندهٔ یک خط ارتباطی (که اغلب لینک نامیده می‌شوند) هستند. برای انتخاب بهترین مسیر بین یک زوج مسیریاب مفروض، یک الگوریتم خاصّ، «کوتاهترین مسیر» بین آن دو را در این گراف پیدا می‌کند.

مفهوم «کوتاهترین مسیر» نیاز به اندکی توضیح دارد: یکی از روشهای محاسبهٔ طول مسیر، شمارش تعداد «گام» است. با استناد به این معیار، مسیرهای ABC و ABE در شکل ۵-۷ طول معادلی دارند. یکی دیگر از معیارها، فاصلهٔ جغرافیایی گره‌ها از یکدیگر برحسب کیلومتر است که در این حالت ABC به وضوح از ABE طولانی‌تر می‌باشد (با فرض آنکه شکل به گونه‌ای ترسیم شده که مقیاسی از نقشهٔ واقعی را ارائه می‌دهد).

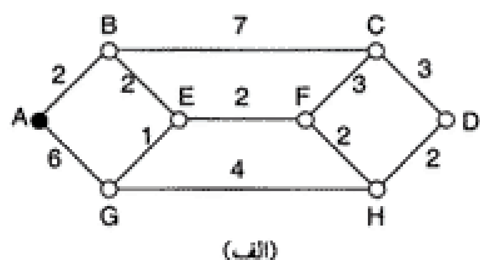
با این وجود، به غیر از معیارهای تعداد گام یا فاصله فیزیکی، معیارهای دیگری نیز وجود دارند. به عنوان مثال هر کمان از گراف می‌تواند برچسبی داشته باشد که مقدار متوسط تأخیر صفّ [صفّ درون حافظه مسیریاب] و تأخیر انتقال را تعیین کند؛ این برچسبها بطور متناوب و با استفاده از برخی بسته‌های استاندارد محاسبه می‌شوند. در چنین گراف برچسب‌داری، کوتاهترین مسیر همانا سریعترین مسیر است (یعنی مسیر با حداقل تاخیر)، نه مسیری که کمترین تعداد کمان یا کوتاهترین فاصلهٔ جغرافیایی را دارد.

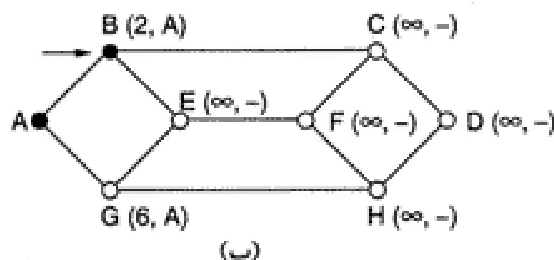
در حالت کلی، برچسب هر کمان می‌تواند برحسب تابعی از پارامترهای «فاصله»، «پهنای باند»، «میانگین ترافیک»، «هزینهٔ ارتباط»، «طول متوسط صف»، «تأخیر اندازه‌گیری‌شده» یا عوامل دیگر، محاسبه شود. با تغییر در «تابع وزن‌دهی» (Weighting Function)، این الگوریتم می‌تواند «کوتاهترین مسیر» را برحسب یکی از این معیارها یا ترکیبی از آنها بدست بیاورد.

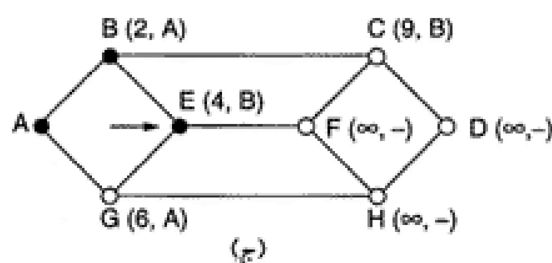
تاکنون الگوریتمهای متعددی برای محاسبهٔ کوتاهترین مسیر بین دو گره از گراف، معرفی شده‌اند. یکی از آنها توسط «دایکسترا» (Dijkstra, 1959) ارائه شد. در الگوریتم دایکسترا، هر گره دارای برچسبی است که فاصلهٔ آن گره را تا یک گره مبداء برحسب بهترین مسیر، مشخص می‌کند. (در شکل ۵-۷ این برچسبها درون پرانتز درج شده‌اند.) در ابتدا هیچ مسیری مشخص نیست فلذا تمام گره‌ها در برچسب خود علامت بی‌نهایت دارند. در حین اجرای الگوریتم و پیدا شدن مسیرها این برچسبها تغییر می‌کنند تا مسیرهای بهتر را مشخص نمایند. هر برچسب می‌تواند یکی از دو حالت «موقت» (Tentative) یا «دائم» (Permanent) داشته باشد. در ابتدای کار تمام برچسبها در حالت موقتی علامت خورده‌اند. پس از آن که مشخص شد یک برچسب کوتاهترین مسیر ممکن را از مبداء تا آن گره مشخص کرده، حالت آن گره به حالت «دائم» تغییر کرده و از آن به بعد هیچگاه عوض نخواهد شد.

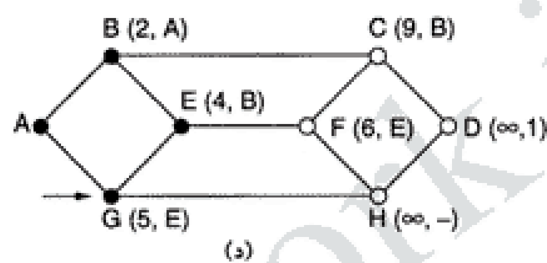
---

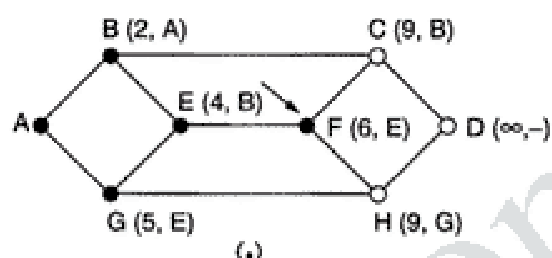
۱. اگر هر یک از مسیریابها، مسیرهای بهینه را متفاوت از دیگری تشخیص بدهند و بر اساس اطلاعات ناقص خود مسیر انتخاب کنند، گاهی حلقهٔ بی‌نهایت ایجاد می‌شود. -م

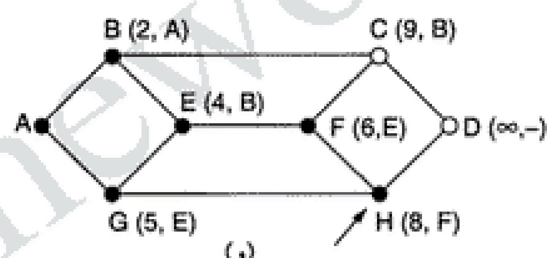












شکل ۵–۷. پنج مرحلهٔ نخست از روال محاسبهٔ کوتاهترین مسیر از A به D. فلشها «گره کار» را مشخص می‌کنند.

برای آن که ببینیم الگوریتم برچسب‌گذاری چگونه کار می‌کند، گراف «بدون جهت» و «وزن‌دار» شکل ۵–۷–الف را در نظر بگیرید که در آن وزنها فرضاً فاصله را مشخص کرده است. می‌خواهیم کوتاهترین مسیر از A به D را پیدا کنیم. کار را با تغییر علامت گره A به حالت «ثابت» (که به صورت یک دایرهٔ توپر مشخص شده) شروع می‌کنیم. سپس به نوبت هر یک از گره‌های مجاور گره A (که گره کار نام دارد) را آزمایش می‌کنیم و برچسبشان را برحسب فاصلهٔ آنها تا A، تغییر می‌دهیم. هرگاه برچسب یک گره را تغییر دادیم، در آن برچسب، نام گره‌ای که آزمایش بر مبنای آن صورت گرفته درج می‌شود تا بعداً بتوان مسیر نهایی را بدست آورد. [یعنی مثلاً وقتی گره‌های همسایهٔ A یعنی B و G را بررسی می‌نماییم و فاصلهٔ آنها تا A را در برچسبشان تغییر می‌دهیم، باید نام گره کار قبلی یعنی A در برچسب آنها درج شود. -م] پس از آزمایش گره‌های مجاور A، تمام گره‌هایی که به صورت موقتی علامت خورده‌اند [یعنی دایره‌های توخالی] را در «**کلّ گراف**» بررسی کرده و گره‌ای با کوچکترین برچسب را انتخاب و آن را به صورت «دائم» علامتگذاری می‌کنیم. (شکل ۵–۷–ب) این گره به عنوان «گره کار» جدید انتخاب می‌شود.

حال از گره B شروع کرده و تمام گره‌های مجاور آن را آزمایش می‌نماییم. اگر مجموع برچسب B و فاصلهٔ B تا گره مورد نظر از آنچه که درون برچسب B نوشته شده کمتر باشد برچسب آن گره تغییر می‌کند چراکه مسیری کوتاهتر را پیدا کرده‌ایم. [به عبارت دیگر وقتی گره‌های همسایهٔ B را بررسی می‌نماییم ـ یعنی E و C را ـ مجموع فاصلهٔ C تا B یعنی ۷ را با فاصله B تا A یعنی ۲ جمع می‌کنیم و چون ۹ از مقدار بی‌نهایت در برچسب C کمتر است برچسب آن به صورت (9,B) تغییر می‌کند یعنی تا رسیدن به گره A فاصله ۹ و گره قبلی B است.-م]

پس از آن که گره‌های مجاور «گره کار» بررسی شدند و در صورت لزوم برچسبهای موقت تغییر نمودند، برای پیدا کردن گره‌ای با علامت موقت و کمترین مقدار فاصله، کل گراف را جستجو می‌نماییم. این گره به حالت «دائم» تغییر وضعیت داده و گره کار جدید در مرحلهٔ بعد خواهد شد. شکل ۵-۷ پنج مرحلهٔ اول این الگوریتم را نشان می‌دهد.

برای آنکه ببینیم چرا این الگوریتم بدرستی کار می‌کند به شکل ۵-۷-ج دقت نمایید. در این مرحله، حالت گره E را به حالت «دائم» تغییر داده‌ایم. فرض کنید که مسیری کوتاهتر به غیر از ABE مثل AXYZE وجود داشته باشد. دو امکان وجود دارد: یا گره Z قبلاً به حالت «دائم» علامت خورده است یا حالت «موقت» دارد. اگر حالت دائم داشته باشد بنابراین نتیجه می‌گیریم که E قبلاً بررسی شده است (در همان مرحله‌ای که گره Z در آن مرحله علامت دائم خورده است) لذا مسیر AXYZE از چشم ما دور نمانده و قاعدتاً نمی‌توانسته کوتاهترین مسیر باشد.

اکنون حالتی را در نظر بگیرید که در آن Z هنوز برچسب موقت دارد. اگر برچسب Z بیشتر یا مساوی آنچه که در برچسب E درج شده، باشد که در این حالت مسیر AXYZE نمی‌تواند از ABE کوتاهتر باشد؛ یا آنکه برچسب Z کوچکتر از برچسب E است که در این حالت Z بجای E علامت دائم می‌خورد و E باید قبل از Z بررسی شود [یعنی از طریق E باید Z بررسی شود و مسیر AXYEZ خواهد بود. -م]

این الگوریتم در شکل ۵-۸ ارائه شده است. متغیرهای سراسری n و dist توصیف‌کنندهٔ گراف هستند و قبل از فراخوانی تابع shortest_path مقداردهی می‌شوند. تنها تفاوت بین این برنامه و الگوریتمی که در بالا تشریح شد آن است که در برنامهٔ شکل ۵-۸ محاسبهٔ کوتاهترین مسیر از آخر (یعنی از گره پایانی) شروع شده است. از آنجایی که کوتاهترین مسیر از t به s در یک گراف بی‌جهت هیچ تفاوتی با کوتاهترین مسیر از s به t ندارد لذا آنکه از کدام گره شروع کنیم اهمیتی ندارد. (مگر آن که کوتاهترین مسیرها بیش از یکی باشند و جستجوی معکوس مسیر دیگری را نتیجه بدهد.) دلیل آنکه جستجوی کوتاهترین مسیر برعکس انجام می‌گیرد آن است که برچسب هر گره با گره قبلی خود (نه گره بعدی) تغییر می‌کند و وقتی مسیر نهایی در متغیر خروجی یعنی path کپی می‌شود، مسیر معکوس خواهد شد [چون عمل کپی از آخر به اول انجام می‌گیرد]. چون از گره آخر به اول شروع کردیم و نهایتاً مسیر بطور معکوس کپی شد این دو، اثر هم را خنثی کرده و جواب نهایی بطور صحیح تولید خواهد شد. [جواب نهایی در متغیر path ، بهترین مسیر از گره مبداء به گره مقصد خواهد بود.]

۵-۲-۳ الگوریتم سیل‌آسا (Flooding)

یکی از الگوریتمهای ایستا در ارسال بسته‌ها، **«الگوریتم سیل‌آسا»** است که براساس آن هر بستهٔ ورودی به یک مسیریاب، بر روی تمام خطوط خروجی (به استثنای خطی که بسته از طریق آن دریافت شده) ارسال می‌شود. این فرآیند سیل‌آسا، به وضوح منجر به تولید تعداد بسیار زیادی بسته‌های تکراری (در حقیقت بی‌نهایت بسته) خواهد شد مگر آنکه برای خاتمهٔ آن، سنجشی صورت بگیرد. یکی از معیارهای سنجش، آنست که در سرآیند هر بسته یک **«شمارندهٔ گام»** (Hop Counter) داشته باشیم و در هر گام یک واحد از آن کم شود و هنگامی که مقدار این شمارنده به صفر رسید بسته حذف شود. حالت آرمانی آن است که مقدار اولیهٔ این شمارنده معادل طول مسیر بین مبداء و مقصد در نظر گرفته شود ولیکن اگر فرستنده نداند که طول مسیر چقدر است می‌تواند مقدار این شمارنده را به بدترین حالت یعنی بزرگترین طول مسیر در شبکه تنظیم نماید.[۱]

راهکار دیگر برای خاتمه دادن به این فرآیند آن است که فهرست بسته‌هایی که قبلاً یکبار به صورت سیل‌آسا ارسال شده‌اند را نگاه داریم تا از ارسال مجدد آن اجتناب شود. برای انجام این کار، مسیریاب مبداء باید در هر بسته

۱. در ادبیات شبکه به بزرگترین طول مسیر، «قطر شبکه» گفته می‌شود.

```
#define MAX_NODES 1024           /* maximum number of nodes */
#define INFINITY 1000000000      /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];       /* dist[i][j] is the distance from i to j */
void shortest_path(int s, int t, int path[])
{ struct state {            /* the path being worked on */
      int predecessor;        /* previous node */
      int length;           /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];
  int i, k, min;
  struct state *p;
  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;           /* k is the initial working node */
  do {          /* Is there a better path from k? */
      for (i = 0; i < n; i++)          /* this graph has n nodes */
            if (dist[k][i] != 0 && state[i].label == tentative) {
                  if (state[k].length + dist[k][i] < state[i].length) {
                        state[i].predecessor = k;
                        state[i].length = state[k].length + dist[k][i];
                  }
            }

      /* Find the tentatively labeled node with the smallest label. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
            if (state[i].label == tentative && state[i].length < min) {
                  min = state[i].length;
                  k = i;
            }
      state[k].label = permanent;
  } while (k != s);

  /* Copy the path into the output array. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

شکل ۵-۸ الگوریتم دایجکسترا برای محاسبهٔ کوتاهترین مسیر در یک گراف.

که از یک ماشین میزبان دریافت می‌کند یک «شماره ترتیب» قرار بدهد. هر مسیریاب نیز باید فهرستی تشکیل بدهد و در آن شماره ترتیب بسته‌هایی را که قبلاً از یک مسیریاب دریافت کرده، درج کند. با ورود هر بسته ابتدا شمارهٔ ترتیب آن درون این فهرست جستجو می‌شود؛ اگر شمارهٔ آن در این فهرست وجود داشته باشد دیگر بهیچوجه (به روش سیل‌آسا) ارسال نخواهد شد (چون قبلاً یکبار ارسال شده است).

برای آنکه از رشد بی‌نهایت این فهرست اجتناب شود در فهرست به ازای هر مسیریاب مبداء، تنها یک شمارنده نگهداری می‌شود و مقدار این شمارنده یعنی k نشان می‌دهد که بسته‌های تا شمارهٔ k قبلاً دریافت شده‌اند. [یعنی در حقیقت فقط شمارهٔ ترتیب آخرین بستهٔ تولید شده توسط یک مسیریاب مشخص می‌شود. -م] وقتی بسته‌ای وارد یک مسیریاب می‌شود آزمون تکراری بودن آن بسیار ساده است [اگر شمارهٔ آن k یا کمتر از k بود تکراری است]؛ اگر بسته تکراری بود حذف می‌شود. بدین ترتیب به فهرست بسته‌های دریافتی با شمارهٔ کمتر از k نیازی نیست و فقط در اختیار داشتن آخرین شماره یعنی k کفایت می‌کند.

گونهٔ دیگری از روش سیل‌آسا که قابلیت اجرایی بیشتری دارد به نام **«روش سیل‌آسا بصورت انتخابی»** (Selective Flooding) مشهور است. در این الگوریتم، مسیریاب تمام بسته‌های دریافتی از یک خط را بر روی تمام خطوط خروجی ارسال نمی‌کند بلکه فقط آنها را بر روی خطوطی ارسال می‌کند که به صورت تخمینی در مسیر صحیحی قرار دارند. هدایت بسته‌هایی که قرار است به سمت غرب بروند به سمت شرق، توجیه چندانی ندارد مگر آنکه توپولوژی زیرشبکه نوع عجیب و ویژه‌ای باشد و مسیریاب از آن به درستی آگاه باشد.

الگوریتم سیل‌آسا در اغلب کاربردها عملی نیست ولیکن در برخی از موارد خاص کاربردهایی دارد. به عنوان مثال در کاربردهای نظامی که گاهی باید برای تعداد زیادی از مسیریابها و در لحظه‌ای کوتاه، حجم زیادی اطلاعات ارسال شود، قدرت روش سیل‌آسا بسیار مطلوب خواهد بود. در پایگاه داده‌های توزیع‌شده (Distributed Database) گاهی لازم است که تمام پایگاههای داده بطور همزمان بهنگام‌سازی شوند؛ در چنین موردی نیز روش سیل‌آسا مفید خواهد بود. در شبکه‌های بی‌سیم تمام پیامهایی که توسط یک ایستگاه ارسال می‌شود توسط بقیهٔ ایستگاههایی که در برد رادیویی آن قرار گرفته‌اند قابل شنود است و در حقیقت نوعی از روش سیل‌آسا محسوب می‌شود و برخی از الگوریتمها از این ویژگی در راستای کار خود بهره می‌گیرند. چهارمین کاربرد روش سیل‌آسا، آنست که می‌توان از آن به عنوان معیاری برای مقایسه با الگوریتمهای دیگر مسیریابی بهره گرفت. در روش سیل‌آسا همیشه کوتاهترین مسیرها انتخاب می‌شود چرا که بسته‌ها از تمام مسیرهای ممکن به صورت موازی ارسال خواهد شد. طبعاً هیچ الگوریتم دیگری نمی‌تواند تأخیر کمتری نسبت به این روش داشته باشد (به شرط آنکه از سربار تحمیل شده توسط خود پروسهٔ سیل‌آسا، قابل چشمپوشی باشد.)

۵-۲-۴ مسیریابی بردار فاصله (Distance Vector Routing)

شبکه‌های کامپیوتری مدرن، عموماً بجای استفاده از روشهای ایستا که در قبل بدانها اشاره‌شد از روشهای پویا بهره می‌گیرند زیرا الگوریتمهای ایستا بار فعلی شبکه را در محاسبات مربوط به بهترین مسیرها دخالت نمی‌دهند. دو الگوریتم پویا یعنی **«مسیریابی بردار فاصله»** (DVR) و **«مسیریابی مبتنی بر حالت لینک»** (Link State Routing) از رایج‌ترین روشهای موجود هستند. در این بخش بطور خاص مروری بر الگوریتم نوع اول (یعنی بردار فاصله) خواهیم داشت و در بخش آتی به مطالعهٔ الگوریتم دوم می‌پردازیم.

الگوریتم «مسیریابی بردار فاصله» (DVR) بدین نحو کار می‌کند: هر مسیریاب جدولی را در خود نگه می‌دارد (به عبارتی یک بردار را) که در آن بهترین فاصله تا هر مسیریاب مقصد [یا به عبارتی کمترین هزینهٔ رسیدن به هر مسیریاب در زیرشبکه] و خطی که برای رسیدن به آن مقصد باید مورد استفاده قرار بگیرد، درج شده است. این جداول با مبادلهٔ اطلاعات بین مسیریابهای همسایه، بهنگام‌سازی می‌شود.

الگوریتمهای مسیریابی بردار فاصله گاهی با نامهای دیگری معرفی می‌شوند که اهم این اسامی عبارتند از «الگوریتم مسیریابی بلمن-فورد توزیع‌شده» (Distributed Bellman-Ford) و «الگوریتم فورد-فوکرسون» (Ford-Fulkerson) که به افتخار نام پژوهشگرانی است که آن را ابداع کرده‌اند. (Bellman, 1957; Ford and Fulkerson, 1962) این روش اولین الگوریتم مسیریابی در ARPANET بود و بعداً نیز با نام RIP (Routing Information Protocol) در اینترنت بکار گرفته شد.

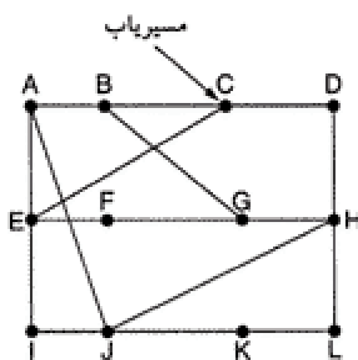
در «مسیریابی بردار فاصله» هر مسیریاب یک جدول مسیریابی دارد که به ازای هر مسیریاب موجود در زیرشبکه، یک «درایه» (Entry) در آن درج شده است. [کل جدول براساس آدرس هر مسیریاب، ایندکس شده است. -م] هر درایه دارای دو بخش است: (۱) خط خروجی مناسب برای رسیدن به مقصد مورد نظر (۲) تخمینی از زمان یا فاصلهٔ رسیدن بدان مقصد. معیار هزینه می‌تواند تعداد گام (Number of Hops) ، تأخیر برحسب میلی‌ثانیه، تعداد کل بسته‌های به صف شده در آن مسیر یا چیزی شبیه به اینها باشد.

فرض بر آن است که هر مسیریاب «فاصلهٔ» خود تا هر یک از همسایه‌هایش را می‌داند. اگر معیار بکار رفته تعداد گام باشد فاصلهٔ هر مسیریاب از همسایه‌هایش دقیقاً ۱ است. اگر معیار، طول صف باشد [یعنی تعداد بسته‌هایی که برای ارسال بر روی یک خط خروجی منتظر هستند] مسیریاب می‌تواند بسادگی هر یک از صفها را بررسی نماید. اگر معیار، تأخیر باشد مسیریاب می‌تواند با ارسال بسته‌های خاصی به نام Echo و دریافت بستهٔ پاسخ (که گیرنده به آن «مُهر زمان» (Timestamp) زده و سریعاً بر می‌گرداند) این تأخیر را مستقیماً اندازه بگیرد.

به عنوان مثال فرض کنید از میزان تأخیر به عنوان معیار بهینگی استفاده شده باشد و مسیریاب تأخیر خود تا هر یک از همسایه‌ها را بداند. هر T میلی‌ثانیه یکبار، تمام مسیریابها برای همسایه‌های خود فهرستی از تأخیر تخمینی در رسیدن به هر یک از مسیریابهای مقصد را ارسال می‌نماید. در ضمن فهرست مشابهی را از همسایه‌های خود دریافت می‌دارد. تصور کنید که یکی از این جداول از طرف مسیریاب X دریافت شود و نماد X_i مشخص کنندهٔ میزان تأخیر برای رسیدن به مسیریاب i باشد. اگر این مسیریاب بداند که تأخیر خودش تا X معادل m میلی‌ثانیه است می‌تواند نتیجه بگیرد که با تأخیر X_i+m میلی‌ثانیه به هر مسیریاب i می‌رسد. با انجام این محاسبه برای تمام جداولی که از همسایه‌ها می‌رسند مسیریاب می‌تواند نتیجه بگیرد که کدام یک از مسیرها بهتر هستند و مقدار تخمینی تأخیر و همچنین خط متناظر با بهترین مسیر را در جدول جدید خود درج می‌نماید. بخاطر داشته که جدول قدیمی هر مسیریاب در محاسبات دخالت داده نمی‌شود.

فرآیند بهنگام‌سازی، در شکل ۵-۹ به تصویر کشیده شده است. قسمت «الف» از شکل، ساختار یک زیرشبکه را نشان می‌دهد. چهار ستون سمت چپ در بخش «ب» شکل، «بردار تأخیر» چهار همسایهٔ مسیریاب J است که در لحظهٔ بهنگام‌سازی دریافت شده است. مسیریاب A ادعا کرده که تا B دوازده میلی‌ثانیه تأخیر دارد؛ ۲۵ میلی‌ثانیه تاخیر تا C ، ۴۰ میلی‌ثانیه تأخیر تا D و به همین ترتیب. فرض کنید مسیریاب J تأخیر خود را تا A و I و H و K به ترتیب مقادیر ۸، ۱۰، ۱۲ و ۶ اندازه‌گیری کرده یا تخمین زده باشد.

حال ببینیم مسیریاب J چگونه مسیرهای جدید برای رسیدن به مسیریاب G را محاسبه می‌کند. J می‌داند که برای رسیدن به A ، ۸ میلی‌ثانیه تأخیر خواهد داشت و A نیز ادعا کرده که قادر است با تأخیر ۱۸ میلی‌ثانیه به G برسد، لذا J متوجه می‌شود که اگر بسته‌های خود را که به مقصد G روانه هستند برای A بفرستد کل تأخیر حدود ۲۶ میلی‌ثانیه [۸ + ۱۸] خواهد بود. به همین طریق، J تأخیر خود تا G از طریق I ، H و K را به ترتیب ۴۱ (۱۰ + ۳۱)، ۱۸ (۱۲ + ۶) و ۳۷ (۶ + ۳۱) میلی‌ثانیه محاسبه می‌نماید. از بین این مقادیر، ۱۸ بهترین است لذا J در جدول مسیریابی خود، درایه‌ای برای G تشکیل می‌دهد و در آن درج می‌کند که تأخیر رسیدن به G ، ۱۸ میلی‌ثانیه است و مسیر مربوطه از طریق H می‌گذرد. [یعنی از آن به بعد هر بسته‌ای که به J وارد شود و بخواهد به G برود به سمت



(الف)

To	A	I	H	K	تخمین تاخیر دیگر مسیریابها تا J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	تاخیرخط JA	تاخیرخط JI	تاخیرخط JH	تاخیرخط JK	جدول مسیریابی جدید در J	
	8	10	12	6		

بردارهای دریافتی از چهار همسایهٔ J

(ب)

شکل ۵-۹. (الف) یک زیرشبکه (ب) جداول دریافتی از A، I، H و K و جدول مسیریابی جدید در J .

مسیریاب H ارسال خواهد شد و تأخیر تخمینی کل مسیر، ۱۸ میلی‌ثانیه خواهد بود. -م]
همین محاسبه برای دیگر مسیریابهای مقصد نیز انجام می‌شود و نهایتاً جدول مسیریابی جدید (نشان داده شده در سمت راست شکل ۵-۹) بدست خواهد آمد.

مشکل شمارش تا بی‌نهایت (Count-To-Infinity Problem)

مسیریابی بردار فاصله از دیدگاه تئوری به درستی کار می‌کند ولیکن در عمل اشکالات جدی خواهد داشت: اگرچه نهایتاً به جواب صحیح همگرا خواهد شد ولی این همگرایی بسیار کند خواهد بود.[1] بالاخص، این الگوریتم به «خبرهای خوب» واکنش سریع نشان می‌دهد در حالی که «خبرهای بد» را به آهستگی منتقل می‌کند![2] یک مسیریاب را در نظر بگیرید که در جدول او تأخیر رسیدن به مقصد X، مقدار بسیار بالا، محاسبه و درج شده است. اگر در مرحلهٔ بعدی مبادلهٔ جداول [یعنی در لحظهٔ بهنگام‌سازی]، مسیریاب همسایهٔ او (مثلاً A)، هزینهٔ کمتری را برای رسیدن به X اعلام کند، او بلافاصله مسیر قبلی خود را تغییر داده و از آن به بعد مسیر ارسال ترافیک به X را از طریق A انتخاب خواهد کرد. در هر بار مبادلهٔ بردارهای هزینه، «خبرهای خوب» بسرعت پردازش می‌شوند.

برای آنکه ببینیم چگونه اخبار خوب منتشر می‌شوند به زیرشبکهٔ پنج گره‌ای (خطی) شکل ۵-۱۰ دقت کنید که در آن معیار تأخیر «تعداد گام» است. فرض کنید A در همان ابتدا از کار افتاده و تمام مسیریابها از این موضوع آگاه هستند. به عبارت دیگر تمام مسیریابها تأخیر خود تا A را به مقدار بی‌نهایت تنظیم کرده‌اند.

وقتی A به کار می‌افتد دیگر مسیریابها با مبادلهٔ بردار [جدول هزینه] بلافاصله از این موضوع آگاه خواهند شد. برای سادگی فرض کرده‌ایم یک ناقوس بزرگ در جایی وجود دارد که بطور متناوب به صدا درمی‌آید تا همهٔ

۱. منظور از همگرایی رسیدن به یک جدول مسیریابی با مقادیر درست، واقعی و پایدار است. -م

۲. منظور از خبرهای خوب کاهش تأخیر در مسیرها، اضافه شدن لینک یا مسیریاب جدید و منظور از خبرهای بد خرابی یک خط، از کار افتادن یک مسیریاب یا افزایش تأخیر است. -م

(الف)

A	B	C	D	E	
	•	•	•	•	Initially در بدو شروع
	1	•	•	•	پس از اولین مبادله جدول
	1	2	•	•	پس از دومین مبادله جدول
	1	2	3	•	پس از سومین مبادله جدول
	1	2	3	4	پس از چهارمین مبادله جدول

(ب)

A	B	C	D	E	
	1	2	3	4	Initially در بدو شروع
	3	2	3	4	پس از اولین مبادله جدول
	3	4	3	4	پس از دومین مبادله جدول
	5	4	5	4	سومین مبادله جدول
	5	6	5	6	چهارمین مبادله جدول
	7	6	7	6	پنجمین مبادله جدول
	7	8	7	8	ششمین مبادله جدول
		⋮			
	•	•	•	•	

شکل ۵-۱۰. مشکل «شمارش تا بی‌نهایت».

مسیریابها بطور همزمان عملیات مبادلهٔ بردارها (جداول) را شروع نمایند!! در اولین مبادله، B آگاه می‌شود که همسایهٔ سمت چپ او تأخیری معادل صفر به A دارد. B در جدول مسیریابی خود یک درایه (Entry) ایجاد و در آن تأخیر رسیدن به A را ۱ درج می‌نماید. هنوز بقیهٔ مسیریابها گمان می‌کنند که A غیرفعال است. تأخیر رسیدن به A در درایه‌های جدول مسیریابی تمام مسیریابها (در این لحظه) در سطر دوم از شکل ۵-۱۰-الف مشاهده می‌شود. در مبادلهٔ بعدی C متوجه می‌شود که B مسیری به A با طول ۱ دارد و به همین دلیل جدول خود را بهنگام می‌کند و در آن مسیری به A از طریق B با طول ۲ را مشخص می‌نماید ولی هنوز D و E از این خبر خوب آگاه نیستند. در یک زیرشبکه، اخبار خوب در هر بار مبادلهٔ جداول، یک گام به جلو منتشر می‌شود. اگر در یک زیرشبکه، طولانی‌ترین مسیر N گام باشد پس از N بار مبادله، همهٔ مسیریابها از اضافه شدن خطوط یا مسیریابهای جدید به زیرشبکه (خبر خوب)، آگاه خواهند شد.

حال به وضعیت شکل ۵-۱۰-ب نگاه کنید که در آن تمام خطوط و مسیریابها در ابتدا فعال هستند. فاصلهٔ مسیریابهای B، C، D و E تا A به ترتیب عبارتند از: ۱، ۲، ۳ و ۴. به ناگاه مسیریاب A از کار می‌افتد یا مثلاً خط بین A و B قطع می‌شود که هر دوی این حالات از دیدگاه B فرقی نمی‌کنند.

پس از اولین مبادلهٔ بسته، B از A چیزی نمی‌شنود. خوشبختانه C می‌گوید: «نگران نباش! من مسیری به A به طول ۲ دارم». B به درستی نمی‌داند که مسیری که C اعلام کرده از خود او (یعنی B) می‌گذرد. آنچه که B حدس زده آنست که شاید C ده خط دیگر دارد و می‌توان از طریق این خطوط با هزینهٔ ۲، به A رسید. در نتیجه، B گمان می‌کند که قادر است از طریق C با هزینهٔ ۳ به A برسد! D و E درایه‌های جدول مسیریابی خود را در اولین مبادله تغییر نمی‌دهند.

در دومین مبادله، C متوجه می‌شود که هر یک از همسایه‌های او ادعا کرده‌اند که مسیری به A با هزینهٔ ۳ دارند لذا یکی از آنها را انتخاب کرده و فاصلهٔ جدید رسیدن به A را به ۴ تنظیم می‌کند. (به گونه‌ای که در سطر سوم از شکل ۵-۱۰-ب نشان داده شده است.) مبادلات بعدی جداول، نتایج نشان داده شده در ادامهٔ شکل ۵-۱۰-ب را تولید خواهد کرد.

از این شکل باید روشن شده باشد که چرا خبرهای بد به کندی منتشر می‌شوند: همهٔ مسیریابها هزینهٔ جدید را یک واحد بیش از نتیجهٔ مینیمم‌گیری از هزینه‌های اعلام شده، در نظر می‌گیرند. [به عبارت دیگر هزینه‌های جدید براساس مقادیری محاسبه می‌شود که قدیمی و اشتباه است. -م] به تدریج مقدار فاصلهٔ درج شده در جدول تمام مسیریابها به سمت بی‌نهایت رشد می‌کند ولیکن تعداد دفعات مبادلهٔ جداول به عددی که برای مقدار «بی‌نهایت» در

نظر گرفته شده بستگی دارد. به همین دلیل بهترین کار آن است که مقدار بی‌نهایت، معادل طول بزرگترین مسیر در زیرشبکه به اضافهٔ ۱ ، در نظر گرفته شود. اگر معیار هزینه، «زمان تأخیر» باشد هیچ حد بالایی را نمی‌توان تعریف کرد مگر آنکه حد بالا را آنقدر زیاد فرض کنیم که با مسیری که بطور طبیعی تأخیر آن بالاست به عنوان مسیر از کار افتاده و خراب رفتار نشود. مشکل فوق‌الذکر به نام **«مشکل شمارش تا بی‌نهایت»** مشهور است. تلاشهایی برای حل این مشکل انجام گرفته (مثل روش split horizon with poisoned reverse که در RFC 1058 تشریح شده) ولی هیچکدام از آنها موفق عمل نکرده‌اند. اصل این مشکل از آنجایی ناشی شده که وقتی X به Y می‌گوید که مسیری به جایی دارد، Y بهیچوجه نمی‌تواند بفهمد که آیا خودش بر روی این مسیر قرار گرفته یا نه!

۵-۲-۵ مسیریابی حالت لینک (Link State Routing)

از «مسیریابی بردار فاصله» تا سال ۱۹۷۹ در ARPANET استفاده می‌شد و در همین ایام به «مسیریابی حالت لینک» (Link State) تغییر کرد. دو مشکل اساسی منجر به زوال آن شد: اول آنکه در این الگوریتم معیار تأخیر، طول صف[1] در نظر گرفته می‌شد و پهنای باند هر یک از خطوط در محاسبات انتخاب بهترین مسیر، دخالت داده نمی‌شد. در ابتدا تمام خطوط 56Kbps بودند لذا پهنای باند خط مورد مهمی نبود ولی پس از آنکه برخی از خطوط به سرعت 230 Kbps و برخی دیگر به 44 Mbps ارتقاء یافتند، بحساب نیاوردن پهنای باند، مشکلی عمده به حساب می‌آمد. البته این امکان وجود داشت که بتوان معیار تأخیر را به پهنای باند خط تغییر داد ولیکن مشکل دیگری ایجاد می‌شد و آن هم اینکه همگرایی الگوریتم بسیار طولانی می‌شد (بدلیل مشکل شمارش تا بی‌نهایت). به همین دلایل با الگوریتم کاملاً جدیدی که اکنون **«مسیریابی حالت لینک»** (LS) نامیده می‌شود، تعویض شد. امروزه گونه‌های متفاوتی از «مسیریابی حالت لینک» مورد استفاده قرار می‌گیرد.

ایدهٔ اصلی و زیربنای مسیریابی حالت لینک (LS) ، ساده است و می‌توان آن را در پنج بند بیان کرد. هر مسیریاب باید به ترتیب زیر عمل کند:

۱. همسایه‌های خود را شناسایی کرده و آدرسهای شبکهٔ آنها را بدست بیاورد.

۲. تأخیر یا هزینهٔ هر یک از همسایه‌های خود را اندازه‌گیری نماید.

۳. بسته‌ای بسازد و اطلاعاتی که از همسایه‌ها کسب کرده ، در آن جاسازی کند.

۴. این بسته را برای تمام مسیریابهای دیگر بفرستد.

۵. کوتاهترین مسیر رسیدن به دیگر مسیریابها را محاسبه نماید.

بدین ترتیب، توپولوژی کامل زیرشبکه و تمام تأخیرها (از طریق آزمایش) اندازه‌گیری شده و بین تمام مسیریابها توزیع می‌شود. برای پیدا کردن کوتاهترین مسیرها به تمام مسیریابهای زیرشبکه، می‌توان از «الگوریتم دایکسترا» بهره گرفت. در ادامه هر یک از این پنج مرحله را به تفصیل بررسی خواهیم نمود.

شناسایی همسایه‌ها

وقتی یک مسیریاب آغاز به کار می‌کند [بوت می‌شود] اولین وظیفهٔ او شناسایی همسایه‌های خودش است. این کار با ارسال یک بستهٔ خاص به نام **«بستهٔ سلام»** (HELLO Packet) بر روی تمام خطوط نقطه‌به‌نقطه انجام می‌شود. انتظار می‌رود که مسیریابهایی که در طرف مقابل هر خط هستند پاسخی برگردانده و خود را معرفی کنند. این نامها [یا به عبارت بهتر آدرس مسیریابها] باید جهانی، منحصر به فرد و یکتا باشند چراکه بعداً وقتی یک مسیریاب راه دور می‌شنود که سه مسیریاب همگی مثلاً به F متصلند، دانستن این موضوع که آیا هر سه F یکی هستند بسیار

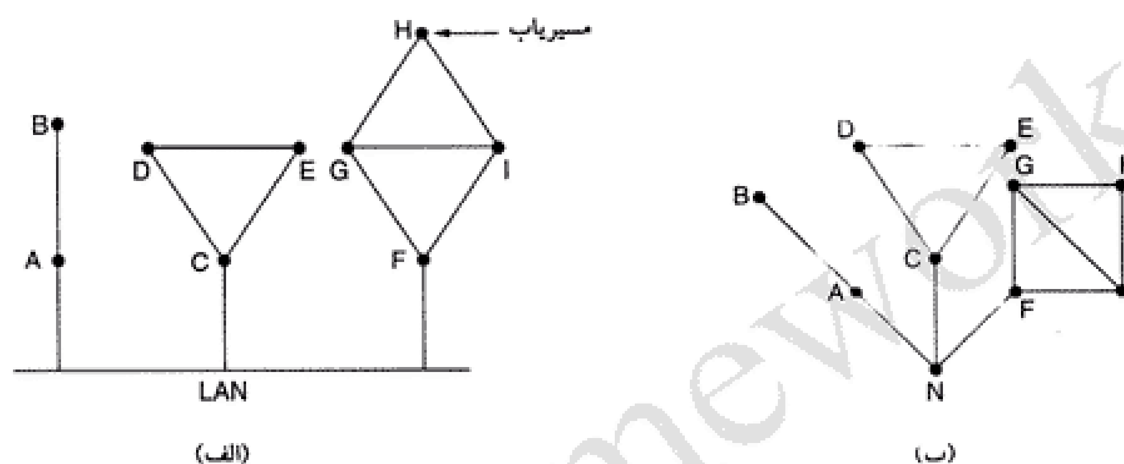
---

۱. تعداد بسته‌های به صف شدهٔ منتظر ارسال بر روی یکی از خطوط خروجی مسیریاب، به طول صف تعبیر می‌شود. -م

حیاتی است.

وقتی دو یا چند مسیریاب از طریق LAN به هم متصل شده باشند وضعیت اندکی پیچیده‌تر است. شکل ۵-۱۱-الف یک شبکهٔ LAN را نشان می‌دهد که سه مسیریاب A، C و F مستقیماً بدان متصل شده‌اند. بگونه‌ای که نشان داده شده این مسیریابها به یک یا چند مسیریاب دیگر نیز متصلند.

یک روش برای مدلسازی LAN آنست که همانند شکل ۵-۱۱-ب، آن را به عنوان یک گره در نظر بگیریم. در اینجا یک گره مصنوعی جدید به نام N معرفی کرده‌ایم که A و C و F بدانها متصل هستند. این واقعیت که رسیدن از A به C فقط از طریق LAN ممکن است با مسیر ANC بیان می‌شود.



شکل ۵-۱۱. (الف) نه مسیریاب و یک شبکهٔ LAN (ب) مدل گراف از شکل الف.

اندازه‌گیری هزینهٔ خط

«الگوریتم حالت لینک» نیازمند آن است که هر مسیریاب از تأخیر هر یک از همسایه‌های خود آگاه باشد (یا حداقل تخمینی معقول از تأخیر آنها داشته باشد.) راه مستقیم اندازه‌گیری این تأخیر آن است که یک بستهٔ خاص به نام Echo بر روی خط مورد نظر ارسال شده و طرف مقابل موظف به برگرداندن فوری آن بسته باشد. با اندازه‌گیری زمان رفت و برگشت این بسته و تقسیم آن بر ۲، مسیریاب فرستنده می‌تواند تخمینی معقول از تأخیر بدست بیاورد. برای بدست آوردن نتایج بهتر می‌توان این آزمون را چندین بار تکرار کرد و میانگین مقادیر اندازه‌گیری‌شده را در نظر گرفت. البته در این روش تلویحاً فرض بر آن گذاشته شده که تأخیرها متقارن[1] است در حالی که ممکن است همیشه اینگونه نباشد.

یک مسئلهٔ جالب آن است که آیا باید در اندازه‌گیری تأخیر، «میزان بار» (load) را نیز به حساب آورد؟ برای وارد کردن «میزان بار» در محاسبات، «تایمر سنجش زمان رفت‌وبرگشت» (Round-Trip Timer) باید زمانی آغاز به کار کند که بستهٔ Echo به **انتهای صف ارسال** وارد می‌شود. برای نادیده گرفتن میزان بار، تایمر زمانی شروع به اندازه‌گیری می‌کند که بستهٔ Echo به سر صف رسیده باشد.

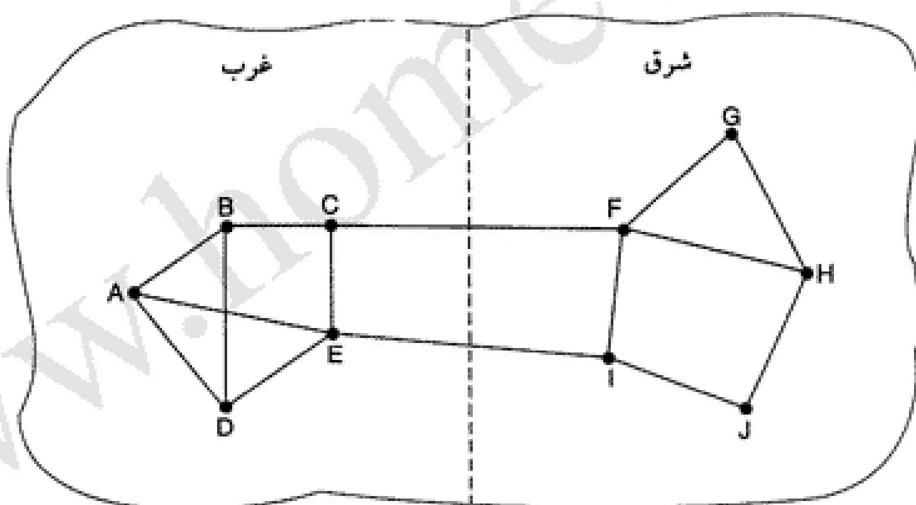
می‌توان در خصوص هر دوی این روشها بحث کرد. در نظر گرفتن تأخیرات ناشی از ترافیک در محاسبات، بدین معنی است وقتی یک مسیریاب می‌خواهد از بین دو خط با پهنای باند مساوی یکی را انتخاب کند (در شرایطی که یکی از آنها برخلاف دیگری با بار سنگین مواجه است)، مسیریاب خطی را که فاقد بار است به عنوان

۱. متقارن بودن تاخیر بدین معناست که تاخیر رسیدن از A به B با تاخیر رسیدن از B به A یکسان است، در حالیکه در بسیاری از محیطها اینگونه نیست. -م

مسیر کوتاهتر در نظر می‌گیرد. چنین انتخابی، کارآیی بهتری را در بر خواهد داشت.

متأسفانه استدلالی بر علیه نظریهٔ «دخالت دادن میزان بار در محاسبهٔ تأخیر» وجود دارد: به زیرشبکه شکل ۵-۱۲ توجه کنید که در آن زیرشبکه به دو بخش شرقی و غربی تقسیم و توسط دو خط CF و EI بهم متصل شده‌اند.

فرض کنید که بیشتر حجم ترافیک بین شرق و غرب از طریق خط CF مبادله شود و در نتیجه این خط با بار سنگین و تأخیر بالایی مواجه است. در نظر گرفتن تأخیر انتظار (یعنی تأخیر صف) در محاسبات کوتاهترین مسیر باعث می‌شود که خط EI جلب توجه نماید. پس از آنکه جدول جدید مسیریابی تنظیم و اعمال شود خط EI به عنوان خط بهینه انتخاب شده و از آن به بعد بخش اعظم ترافیک شرق به غرب از طریق EI عبور خواهد کرد و طبعاً این خط با بار سنگین مواجه خواهد شد. طبعاً در بهنگام‌سازی بعدی، مجدداً خط CF به عنوان مسیر بهینه شناخته می‌شود. در نتیجه ممکن است جدول مسیریابی بطور نوسانی تغییر کرده و منجر به مسیریابی نامنظم شده و اشکالات بالقوهٔ فراوانی تولید کند. اگر از میزان بار چشمپوشی شده و فقط پهنای باند در نظر گرفته شود این مشکل رخ نخواهد داد. البته می‌توان بار را بر روی هر دو خط توزیع کرد ولیکن در این راه حل از مسیر بهینه، استفادهٔ کامل نخواهد شد. علیرغم این، برای اجتناب از بروز تغییرات نوسانی در انتخاب بهترین مسیر، شاید توزیع بار بر روی چند خط راهکاری معقول باشد (البته باید کسری از بار که بر روی هر خط توزیع می‌شود از قبل تعیین شده باشد).

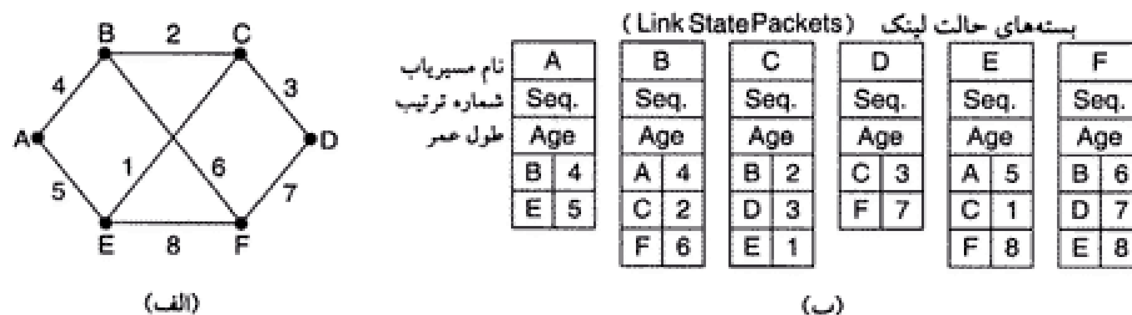


شکل ۵-۱۲. زیرشبکه‌ای که در آن دو بخش شرقی و غربی با دو خط به یکدیگر متصل شده‌اند.

ساخت بسته‌های حالت لینک (Building Link State Packet)

به محض آنکه اطلاعات لازم جهت مبادله گردآوری شد، گام بعدی هر مسیریاب، ساختن بسته‌ای است که این داده‌ها را در بر بگیرد.[1] در ابتدای هر بسته ، هویت فرستندهٔ آن درج می‌شود، سپس فیلدهای «شماره ترتیب» و «طول عمر» (Age) بسته می‌آید (در مورد طول عمر بسته در ادامه توضیح خواهیم داد)؛ در آخر نیز فهرست همسایه‌ها و تأخیر رسیدن بدانها، مشخص می‌شود. در شکل ۵-۱۳-الف یک زیرشبکهٔ نمونه نشان داده شده که در آن، تأخیرها به صورت برچسب عددی، بر روی هر خط مشخص شده‌اند. بسته‌های LS ایجاد شده در هر یک

۱. «بسته‌های حالت لینک» را یک ساختمان دادهٔ استاندارد همانند یک استراکچر در زبان C در نظر بگیرید. از این به بعد این بسته‌ها را بسته‌های LS می‌نامیم. -م



شکل ۵-۱۳. (الف) یک زیرشبکه (ب) بسته‌های حالت لینک (LS) برای این زیر شبکه.

از شش مسیریاب ، در شکل ۵-۱۳-ب مشاهده می‌شود.

ساخت بسته‌های LS ساده است. مشکل‌ترین بخش قضیه تعیین زمانی است که مسیریابها باید به ساختن این بسته‌ها اقدام کنند. یک راه آن است که بطور متناوب و در فواصل زمانی مشخص ایجاد شوند. راه دیگر آن است که مسیریاب فقط زمانی مبادرت به ساخت این بسته ها کند که اتفاق خاصی رخ بدهد، مثلاً یکی از خطوط یا مسیریابهای همسایه از کار بیفتد یا مجدداً فعال شود یا ویژگیهای آن بطور محسوسی تغییر کند. [منظور از ویژگی، میزان تأخیر، بار، پهنای باند یا نظایر آنهاست. -م]

توزیع بسته‌های «حالت لینک»

ظریفترین بخش این الگوریتم، توزیع مطمئن بسته‌های LS است. به محض آنکه بسته‌ها، توزیع و در جداول اعمال شدند، آن مسیریاب که اول از همه این بسته‌ها را دریافت کند مسیرهای خود را تغییر می‌دهد [در حالیکه ممکن است بقیه هنوز چنین کاری نکرده باشند]. در نتیجه مسیریابهای متفاوت ممکن است برای لحظاتی از نسخه‌های متفاوتی از توپولوژی زیرشبکه استفاده کنند که این مسئله منجر به مشکلاتی در مسیریابی صحیح، بروز حلقهٔ بینهایت، از دسترس خارج شدن برخی از ماشینها و مشکلات دیگر شود.

در ابتدا به تشریح الگوریتم اساسی توزیع می‌پردازیم و در ادامه اصلاحاتی را بر روی آن انجام خواهیم داد. ایدهٔ اصلی آن است که برای توزیع بسته‌های LS از روش «سیل آسا» استفاده شود. برای آنکه کنترل این فرآیند را در دست داشته باشیم هر بسته دارای شمارهٔ ترتیب است؛ به ازای تولید هر بستهٔ جدید، به این شمارهٔ ترتیب یک واحد اضافه می‌شود. هرگاه مسیریابها بسته‌ای از این نوع را دریافت کنند زوج آیتم «آدرس مسیریاب مبداء , شمارهٔ ترتیب» آنرا در جایی ذخیره می‌کنند. وقتی یک بستهٔ LS جدید وارد شود ابتدا بررسی می‌شود که آیا این بسته قبلاً نیز دریافت شده است. اگر بسته جدید بود بر روی تمام خطوط خروجی (به استثنای خطی که از روی آن دریافت شده) ارسال می‌شود ولیکن اگر تکراری بود نادیده گرفته می‌شود. اگر بسته‌ای دریافت شود و شمارهٔ ترتیب آن از بزرگترین شماره‌ای که تاکنون دریافت شده، کوچکتر باشد به عنوان بستهٔ قدیمی تلقی و در نظر گرفته نخواهد شد چرا که مسیریاب، نسخهٔ جدیدتری از آن را در اختیار دارد.

این الگوریتم چند مشکل دارد ولی این مشکلات قابل مدیریت و کنترل هستند: اول آنکه اگر شمارهٔ ترتیب با افزایشهای متوالی، در جایی به صفر برگردد مشکلاتی بروز خواهد کرد. راه حل آن است که از یک شماره ترتیب ۳۲ بیتی استفاده شود؛ در این صورت اگر در هر ثانیه یک بستهٔ LS ارسال شود، ۱۳۷ سال طول می‌کشد تا این عدد به صفر برگردد لذا احتمال بروز چنین مشکلی قابل چشمپوشی است!

دوم آنکه اگر یکی از مسیریابها از کار بیفتد روند شمارهٔ ترتیب بسته‌های خود را از دست خواهد داد و اگر بعد از راه اندازی مجدّد، شمارهٔ ترتیب را از صفر شروع کند بسته‌های ارسالی او تکراری تلقی شده و

در نظر گرفته نمی‌شود.

سوم آنکه اگر شمارهٔ ترتیب به نحوی دچار خطا شود و مثلاً در اثر یک بیت خطا شمارهٔ ۴ به ۶۵۵۴۰ تبدیل شود از آن به بعد بسته‌های ۵ تا ۶۵۵۴۰ به عنوان بستهٔ قدیمی دور انداخته خواهد شد چرا که گمان می‌رود که شمارهٔ فعلی ۶۵۵۴۰ است.

راه حل تمام این مشکلات آن است که پس از شمارهٔ ترتیب، طول عمر بسته نیز درج گردد و با گذشت هر ثانیه یک واحد از آن کم شود. وقتی عمر بسته به صفر برسد اطلاعاتی که از آن مسیریاب دریافت شده باید حذف گردد. در یک روال طبیعی (مثلاً هر ده ثانیه یکبار)، بسته‌های جدید LS وارد مسیریاب شده و طبعاً اطلاعات قبلی بطور منظم و قبل از انقضای مهلت اعتبارشان تازه‌سازی می‌شوند؛ بدین ترتیب اعتبار اطلاعات مربوط به هر مسیریاب زمانی منقضی خواهد شد که آن مسیریاب از کار بیفتد (یا آنکه مثلاً در اثر هر گونه رخداد، متوالیاً شش بستهٔ LS از آن مسیریاب دریافت نشود). فیلد «طول عمر» (Age) در خلال فرآیند ارسال سیل‌آسا و توسط هر مسیریاب نیز یک واحد کاهش می‌یابد تا مطمئن شویم که هیچ بسته‌ای نمی‌تواند بطور نامحدودی زنده و در زیرشبکه سرگردان بماند. (هر گاه عمر بسته صفر شود، حذف خواهد شد.)

چند اصلاح در این الگوریتم، آنرا قدرتمندتر خواهد کرد. وقتی یک بستهٔ LS برای ارسال به روش سیل‌آسا وارد یک مسیریاب می‌شود بلافاصله در صف ارسال وارد نخواهد شد بلکه ابتدا بـه یک «فضای انـتظار» وارد می‌شود تا برای مدتی کوتاه، منتظر بماند. اگر قبل از ارسال بستهٔ فعلی، بستهٔ مشابه دیگری از همان مبداء وارد شود، شمارهٔ ترتیب این دو بسته مقایسه شده و در صورت مساوی بودن، بستهٔ تکراری حذف خواهد شد و در صورت تکراری نبودن از بستهٔ قدیمی‌تر صرفنظر می‌شود. برای پیشگیری از خطاهای احتمالی بر روی خطوط مستقیم بین دو مسیریاب، دریافت تمام بسته‌های LS ورودی، تصدیق (Ack) خواهد شد.[۱] وقتی یکی از خطوط خروجی مسیریاب آزاد گردد «فضای انتظار» به روش Round Robin (نوبت چرخشی) پویش می‌شود تا یک بستهٔ LS یا یک بستهٔ اعلام وصول (یعنی Ack) برای ارسال انتخاب گردد.

در جدول شکل ۵-۱۴ ساختمان دادهٔ مورد استفاده در مسیریاب B که برای زیرشبکهٔ شکل ۵-۱۳-الف ایجاد شده، دیده می‌شود. در این جدول، هر سطر متناظر با یک بستهٔ LS تازه وارد است که هنوز بطور کامل پردازش نشده است. در این جدول مبداء هر بسته، شمارهٔ ترتیب، طول عمر آن و مقداری داده ثبت می‌شود. بعلاوه به ازای هر یک از سه خط متصل به B (یعنی خطوط A و C و F) دو پرچم «ارسال» و «اعلام وصول»[۲] در نظر گرفته شده است. «پرچم ارسال» بدین معناست که بسته باید بر روی خط مشخص شده، ارسال شود. «پرچم اعلام وصول» بدین معناست که دریافت این بسته باید بر روی خط مربوطه، به آگاهی طرف مقابل برسد.

از جدول شکل ۵-۱۴، مشخص است که یک بستهٔ LS مستقیماً از A دریافت شده و طبق آنچه که بیتهای پرچم نشان می‌دهند این بسته باید برای C و F ارسال شود و ضمناً وصول آن به A اعلام گردد. به طور مشابه بسته‌ای که از F آمده باید به A و C نیز ارسال شده و دریافت آن به F اعلام گردد.

با این حال شرایط بستهٔ سوم که از E می‌رسد متفاوت است. این بسته دو بار، یکبار از طریق EAB و یکبار از طریق EFB دریافت شده است. در نتیجه بنحوی که بیتهای پرچم نشان می‌دهند، این بسته باید برای C ارسال شود ولیکن فقط کافی است وصول آن به A و F اعلام گردد.

اگر بسته‌ای تکراری دریافت شود در حالی که نسخهٔ اصلی آن هنوز در بافر موجود است، بیتهای پرچم باید تغییر کنند. به عنوان مثال اگر بستهٔ متعلق به C قبل از آنکه مطابق با درایهٔ چهارم (4th Entry) برای A و F ارسال

---

۱. دریافت هر بستهٔ LS به مسیریاب همسایه (که آنرا فرستاده) اعلام خواهد شد نه مبداء آن. -م

۲. Send Flag and Achnowledgement Flag

Source (مبدا)	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

شکل ۵-۱۴. بافر بسته‌های LS برای مسیریاب B در شکل ۵-۱۳.

شود یکبار دیگر از طریق F دریافت گردد، شش بیت پرچم به 100011 تغییر خواهد کرد بدین معنا که باید وصول بسته به F اعلام شود ولی لازم نیست خود بسته برای F ارسال گردد.

محاسبهٔ مسیرهای جدید

به محض آنکه مسیریاب مجموعهٔ کامل بسته‌های LS را گردآوری کرد می‌تواند گراف کل زیرشبکه را تشکیل بدهد چرا که همهٔ لینکها و هزینهٔ آنها مشخص شده است. هر لینک در حقیقت دو بار معرفی شده است، یکبار در هر جهت. [چراکه مثلاً لینک بین A و B، یکبار توسط A و یکبار توسط B شناسایی و معرفی می‌شوند. -م]

حال می‌توان الگوریتم دایکسترا را به صورت محلی اجرا کرد و کوتاهترین مسیر ممکن به تمام نقاط مقصد در زیرشبکه را بدست آورد.[1] نتیجهٔ این الگوریتم می‌تواند در جدول مسیریابی درج شده و مسیریابها عملکرد طبیعی خود را از سر بگیرند.

برای زیرشبکه‌ای با n مسیریاب که هر کدام k همسایه دارند، حافظه لازم برای ذخیرهٔ داده‌های جدول مسیریابی، متناسب با nxk است که در زیرشبکه‌های بزرگ می‌تواند مشکل ایجاد کند. در ضمن زمان محاسبه ، نیز می‌تواند مسئله‌ساز باشد ولیکن علیرغم همهٔ اینها «الگوریتم مسیریابی حالت لینک» (Link State Routing) در محیطهای واقعی بخوبی کار می‌کند.

با این حال، هرگونه اشکال سخت‌افزاری یا نرم‌افزاری می‌تواند مشکلات بسیار جدی برای این الگوریتم به بار بیاورد. (هم‌چنین در عملکرد الگوریتم دیگر مسیریابها نیز مشکل ایجاد می‌کند.) به عنوان نمونه، اگر یک مسیریاب ادعا کند که خطی در اختیار دارد در حالی که نداشته باشد یا فراموش کند خطی را که در اختیار دارد اعلام کند، گراف زیرشبکه صحیح نخواهد بود. همچنین اگر یک مسیریاب در ارسال بسته‌های LS مشکل بهم بزند یا آنها را در حین ارسال خراب کند مشکلاتی جدّی بروز خواهد کرد. سرانجام آنکه اگر حافظهٔ مسیریاب سرریز شود یا نتیجهٔ محاسبات مسیر اشتباه باشد رخدادهای ناگواری در مسیریابی بسته‌ها بوقوع می‌پیوندد. هر گاه رشد زیرشبکه به دهها، صدها یا حتی هزاران گره برسد احتمال از کار افتادن ناگهانی یک یا چند مسیریاب را نمی‌توان نادیده گرفت. تنها راه ممکن آن است که تلاش شود تا در هنگام بروز چنین رخدادهایی میزان آسیبها و مشکلات در حد محدودی کنترل شود. پرلمن (Perlman, 1988) این مشکلات و راه حلهای آنها را تشریح کرده است.

روش «مسیریابی حالت لینک» بطور گسترده‌ای در شبکه‌های واقعی بکار گرفته شده است لذا مختصری در

۱. اجرای الگوریتم دایکسترا به صورت محلی بدین معناست که هر یک از مسیریابها خودشان مستقل از دیگری آن را اجرا می‌کنند. -م

مورد چند پروتکل نمونه که از آن بهره گرفته‌اند خالی از لطف نیست. پروتکل OSPF که در اینترنت بسیار رایج است از همین الگوریتم استفاده می‌کند. در بخش ۵ـ۶ـ۴، OSPF را تشریح خواهیم کرد.

یکی دیگر از «پروتکلهای مبتنی بر حالت لینک»، پروتکل IS-IS[1] است که توسط DECnet طراحی شد و بعداً توسط ISO جهت بکارگیری در کنار پروتکل بی‌اتصال لایهٔ شبکهٔ خود یعنی CLNP، مورد پذیرش و تأیید قرار گرفت. البته بعداً در آن تغییراتی ایجاد شد تا بتواند با پروتکلهای دیگری مثل IP نیز کار کند. پروتکل IS-IS در بخشهایی از ستون فقرات شبکهٔ اینترنت (مثلاً در ستون فقرات قدیمی متعلق به NSFNET) و در برخی از سیستمهای دیجیتال سلولی مثل CDPD بکار رفته است. شرکت Novell Netware نیز گونهٔ ساده‌تری از IS-IS را برای هدایت بسته‌های IPX خود بکار گرفته است.

IS-IS اساساً تصویری از توپولوژی مسیریاب را در زیرشبکه توزیع می‌کند و از طریق آن کوتاهترین مسیرها محاسبه می‌گردد. هر مسیریاب در هنگام اعلام اطلاعات مربوط به «حالت لینک» (LS) مشخص می‌کند که به چه آدرسهایی مستقیماً دسترسی دارد. (آدرسها مربوط به لایهٔ شبکه و جهانی هستند). این آدرسها می‌توانند IP، IPX، AppleTalk یا هر آدرس دیگر باشند. IS-IS همچنین می‌تواند بطور همزمان از چندین پروتکل لایهٔ شبکه پشتیبانی نماید.

بسیاری از نوآوریهایی که در IS-IS طراحی شده بود بعداً مورد پذیرش و استفادهٔ OSPF قرار گرفت. (OSPF چندین سال پس از IS-IS ابداع شد.) برخی از اینها عبارت بودند از روشی برای خاتمهٔ خودکار فرآیند ارسال بسته‌ها به روش سیل‌آسا، مفهوم «مسیریاب برگزیده» (Designated Router) در شبکهٔ LAN و روش محاسبه و پشتیبانی از تقسیم مسیر و همچنین تعریف معیارهای چندگانهٔ هزینه. در نتیجه اختلاف ناچیزی بین IS-IS و OSPF وجود دارد. مهمترین اختلاف این دو آن است که IS-IS به گونه‌ای بسته‌های LS را تعریف و کد کرده است که بسادگی و بطور همزمان می‌تواند اطلاعاتی در خصوص چندین پروتکل لایهٔ شبکه با خود حمل کند، خصوصیتی که OSPF فاقد آن است. این حسن در محیطهای بزرگ که با چندین پروتکل مختلف کار می‌کنند بسیار ارزشمند است.

۵ـ۲ـ۶ مسیریابی سلسله‌مراتبی (Hierachical Routing)

با رشد اندازهٔ شبکه، جداول مسیریابی به همان نسبت رشد می‌کنند. جداول مسیریابی رو به رشد، نه تنها حافظهٔ بیشتری مصرف می‌کنند بلکه به زمان CPU بیشتری برای پویش و جستجوی این جدول و همچنین پهنای باند زیادتری برای ارسال بسته‌های LS (بسته‌های گزارش وضعیت لینک) نیاز خواهد بود. ممکن است رشد شبکه بدان حد برسد که دیگر امکان نگهداری یک «درایه» (Entry) به ازای هر مسیریاب، در جدول مسیریابی وجود نداشته باشد و مسیریاب مجبور به مسیریابی سلسله‌مراتبی (همانند شبکهٔ تلفن) شود.

وقتی از مسیریابی سلسله‌مراتبی استفاده می‌شود، مسیریابها به تعدادی «منطقه» (Region) تقسیم می‌شوند؛ هر مسیریاب تمام جزییات منطقهٔ خود و مسیرهای دقیق رسیدن به هر مقصد در منطقهٔ خود را می‌داند ولی چیزی در خصوص ساختار داخلی مناطق دیگر نمی‌داند. وقتی شبکه‌های مختلف بهم متصل می‌شوند طبیعی است که هر یک از آنها را به عنوان مناطق مجزا در نظر بگیریم تا مسیریابهای درون یک منطقه از دانستن ساختار توپولوژی ناحیهٔ دیگر بی‌نیاز و فارغ باشند.

در شبکه‌های عظیم ممکن است سلسله‌مراتب دو سطحی کفایت نکند. ممکن است نیاز باشد که هر «منطقه» (Region) به تعدادی «دسته» (Cluster)، هر «دسته» به تعدادی «ناحیه» (Zone) و هر ناحیه به تعدادی «گروه»

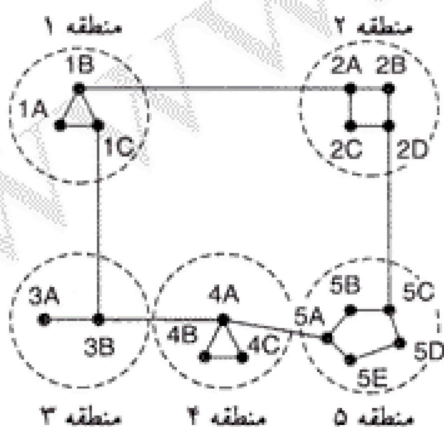
---

۱. Intermediate System-Intermediate System

(Group) تقسیم شود و به همین ترتیب ادامه یابد تا جایی که برای تقسیم‌بندی بیشتر اسمی باقی نماند!! به عنوان مثالی از سلسله‌مراتب چندسطحی، به چگونگی هدایت یک بسته از برکلی کالیفرنیا به مالیندی در کنیا دقت کنید. مسیریاب واقع در برکلی جزییات توپولوژی زیرشبکه خود در کالیفرنیا را می‌داند ولیکن تمام ترافیک متعلق به خارج از ایالت کالیفرنیا را به مسیریاب لوس آنجلس می‌فرستد. مسیریاب واقع در لوس آنجلس تنها قادر به مسیریابی ترافیک داده‌ها بین مسیریابهای داخل کشور است و هر گونه ترافیک خارجی را به نیویورک می‌فرستد. مسیریاب واقع در نیویورک به گونه‌ای برنامه‌ریزی شده که این بسته را به سمت مسیریابی که در کشور مقصد مسئول دریافت ترافیک خارجی است (مثلاً مسیریاب نایروبی) بفرستد. نهایتاً این بسته در کنیا به همین سیاق مسیر خود را ادامه می‌دهد تا به مالیندی برسد.

در شکل ۵-۱۵ یک نمونهٔ کمّی از مسیریابی سلسله‌مراتبی دو سطحی با پنج منطقه را ارائه کرده است. مطابق با شکل ۵-۱۵-ب جدول کامل مسیریابی در مسیریاب 1A دارای ۱۷ «درایه» است. وقتی مسیریابی به صورت سلسله‌مراتبی انجام می‌شود به ازای هر مسیریاب محلی یک درایه در جدول مسیریابی درج می‌شود ولیکن مناطق دیگر فقط در یک مسیریاب خلاصه می‌شوند لذا کل ترافیک مربوط به منطقهٔ ۲ از خط 1B-2A عبور خواهد کرد و مابقی ترافیک خارجی از طریق 1C-3B هدایت خواهد شد. مسیریابی سلسله‌مراتبی، تعداد درایه‌های جدول مسیریابی 1A را از ۱۷ به ۷ تا کاهش می‌دهد. [۳ تا برای مسیریابهای داخل منطقه و ۴ تا به ازای مناطق دیگر] هر چه نسبت تعداد مناطق به تعداد مسیریابهای درون هر منطقه افزایش یابد میزان صرفه‌جویی در فضای جدول بیشتر خواهد بود.

متأسفانه صرفه‌جویی در فضای حافظه رایگان بدست نمی‌آید. بهایی که باید برای آن پرداخت، افزایش طول مسیرها [یا به عبارتی عدم بهینگی کامل مسیرها] است. به عنوان مثال بهترین مسیر از 1A به 5C از منطقهٔ ۲ می‌گذرد در حالی که در مسیریابی سلسله‌مراتبی مسیر تمام ترافیک متعلق به ناحیهٔ ۵ از منطقهٔ ۳ تعیین شده چرا که برای



(الف)

جدول کامل مسیریابی 1A

(مقصد) Dest.	Line	Hops (گام)
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(ب)

جدول مسیریابی سلسله‌مراتبی 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(ج)

شکل ۵-۱۵. مسیریابی سلسله‌مراتبی.

بیشتر مسیریابهای مقصد واقع در ناحیهٔ ۵، مسیر عبوری از منطقهٔ ۲ مناسبتر است.

سؤال قابل توجه آن است که وقتی یک شبکهٔ واحد، بشدت رشد می‌کند، سلسله‌مراتب باید چندسطحی باشد؟ به عنوان مثال زیرشبکه‌ای با ۷۲۰ مسیریاب را در نظر بگیرید. اگر سلسله مراتب وجود نداشته باشد هر مسیریاب در جدول مسیریابی خود به ۷۲۰ درایه نیاز خواهد داشت. اگر زیرشبکه به ۲۴ منطقه و ۳۰ مسیریاب در هر منطقه تقسیم شود، هر مسیریاب به ۳۰ درایه برای مسیریابهای محلی خود به علاوه ۲۳ درایه برای مناطق خارجی، نیاز دارد و بدین ترتیب جدول جمعاً ۵۳ درایه خواهد داشت. اگر سلسله مراتب، سه‌سطحی انتخاب شده و جمعاً هشت «دسته» (Cluster)، هر دسته شامل ۹ ناحیه و در هر ناحیه ۱۰ مسیریاب تعریف شده باشد، هر مسیریاب در جدول خود به ۱۰ درایه برای مسیریابهای محلی خود، ۸ درایه برای مسیریابی در نواحی داخل دستهٔ خودش و ۷ درایه برای دسته‌های خارجی (Distant Cluster) یعنی جمعاً ۲۵ درایه نیاز دارد. دو پژوهشگر بنامهای Kamoun و Kleinrock (۱۹۷۹) پی بردند که بهترین تعداد سطوح سلسله‌مراتب در زیرشبکه‌ای با N مسیریاب، معادل Ln(N) است و هر مسیریاب به جمعاً e.ln(N) عدد درایه نیاز خواهد داشت. همچنین نشان دادند که افزایش طول مؤثر مسیرها یا به عبارت دیگر کاهش بهینگی مسیرها، که از مسیریابی سلسله‌مراتبی منشاء می‌گیرد بقدر کافی کم و نتیجه معمولاً قابل قبول است.

۵-۲-۷ مسیریابی فراگیر (Broadcost Routing)

در برخی از کاربردها، ماشینهای میزبان نیازمند ارسال پیام به همهٔ ماشینهای شبکه یا تعدادی از آنها هستند. به عنوان مثال برای خدمات توزیع گزارشات هواشناسی، بهنگام‌سازی اطلاعات بازار سهام یا برنامه‌های زندهٔ رادیویی، راهکار مناسبتر آنست که داده‌ها را بصورت پخش فراگیر برای تمام ماشینها ارسال کرده و آنها را آزاد بگذاریم تا در صورت تمایل بسته‌های داده را دریافت کرده و بخوانند. «ارسال همزمان یک بسته به تمام ماشینهای مقصد، اصطلاحاً پخش فراگیر (Broadcasting) نامیده می‌شود و راهکارهای متنوعی برای اجرای آن پیشنهاد شده است.»

یکی از روشهای پخش فراگیر که به هیچ ویژگی خاصّی از زیرشبکه نیاز ندارد آن است که ماشین مبداء هر بستهٔ خود را بطور جداگانه برای یکایک ماشینهای مقصد بفرستد. این روش نه تنها پهنای باند را تلف خواهد کرد بلکه ماشین مبداء نیازمند آن است که فهرست کاملی از تمام ماشینهای مقصد در اختیار داشته باشد. اگرچه گاهی در عمل این روش تنها راه ممکن است ولیکن روش چندان مطلوبی نیست.

روش ارسال سیل‌آسا، نامزد دیگری برای پخش فراگیر محسوب می‌شود. هر چند روش سیل‌آسا، راهکار نامناسبی برای شبکه‌های نقطه‌به‌نقطه است ولیکن برای ارسال فراگیر می‌توان بر روی آن حساب باز کرد، خصوصاً وقتی که هیچیک از راهکارهایی که در ادامه معرفی می‌شوند عملی نباشد. استفاده از روش ارسال سیل‌آسا همان مشکلی را دارد که در الگوریتم مسیریابی نقطه‌به‌نقطه به آن اشاره کردیم: یعنی بسته‌های بسیار زیاد تولید می‌شود و پهنای باند بسیار وسیعی مصرف و تباه خواهد شد.

الگوریتم سوم «**مسیریابی چندمقصدی**» (Multidestination Routing) نام دارد. اگر از این روش استفاده شود هر بسته فهرستی از کلیهٔ مقصدهای مورد نظر یا «**نقشه‌ای بیتی**» (Bitmap) از این مقصدها با خود حمل می‌کند. هر گاه بسته‌ای به یک مسیریاب برسد، آن مسیریاب فهرست مقصدهای بسته را بررسی می‌کند تا مجموعهٔ خطوط خروجی منتهی به هر مقصد تعیین شوند. (هر خط خروجی که حداقل به یکی از مقاصد مورد نظر منتهی شود، انتخاب می‌گردد.) مسیریاب به ازای هر یک از خطوط خروجی تعیین شده، یک نسخهٔ جدید از آن بسته را تولید کرده و در آن بسته فقط آدرس مقصدهایی را قرار می‌دهد که مسیر آنها بر روی خط خروجی مربوطه است. [بقیهٔ آدرسهای مقصد حذف می‌شوند چرا که برای دیگر مقصدها خط خروجی جدا انتخاب و نسخه‌ای جداگانه تولید می‌شود. -م] در نتیجه مجموعهٔ آدرسهای مقصد هر بسته بر روی خطوط جداگانه تقسیم و توزیع می‌شود.

پس از چندین گام، هر بسته فقط یک آدرس مقصد با خود حمل می‌کند و با آن به عنوان یک بستهٔ معمولی رفتار می‌شود. «مسیریابی چندمقصدی» همانند بسته‌های مستقل و با آدرس جداگانه عمل می‌کنند، با این تفاوت که بجای ارسال چندین بسته بر روی یک مسیر مشابه، فقط یک بسته ارسال و بدین نحو فقط هزینهٔ ارسال یک بسته پرداخت می‌شود. [در این روش، هر بسته با رسیدن به یک مسیریاب به تعداد کافی تکثیر می‌شود. -م]

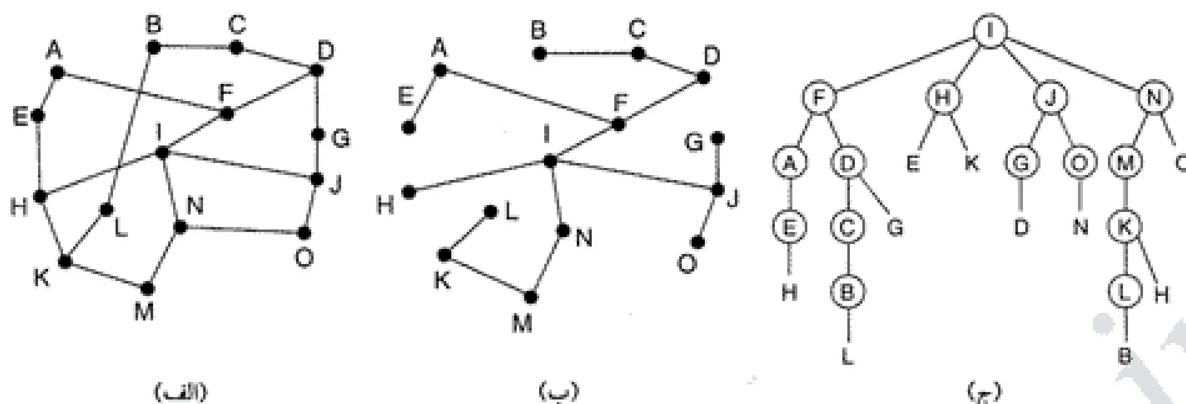
الگوریتم چهارم برای پخش فراگیر آن است که مسیریاب آغازگر این فرآیند، از درخت Sink Tree یا هر درخت پوشای مناسب (Spanning Tree) استفاده نماید. یک **«درخت پوشا»** زیر مجموعه‌ای از کلّ زیرشبکه است که تمام مسیریابها را در برمی‌گیرد ولیکن هیچ حلقه‌ای در آن نیست. اگر هر مسیریاب بداند که کدامیک از خطوط او در «درخت پوشا» قرار گرفته‌اند، می‌تواند یک بستهٔ فراگیر ورودی را فقط بر روی خطوطی در خروجی، بفرستد که بر روی این درخت پوشا هستند. (البته به استثنای خطی که بسته از آن دریافت شده است.) بهره‌وری پهنای باند در این روش بسیار عالی است و برای انجام کار، کمترین بستهٔ لازم تولید خواهد شد. تنها مشکل این روش آن است که هر مسیریاب برای بکارگیری این روش باید دانش و اطلاعات کافی در خصوص درختهای پوشا در زیرشبکه داشته باشد. برخی اوقات این اطلاعات در دسترس هستند (مثلاً در مسیریابی حالت لینک ـ LS ـ) ولی گاهی موجود نیستند (مثل مسیریابی بردار فاصله ـ DV ـ)

آخرین الگوریتم پخش فراگیر، آنست که حتی وقتی مسیریاب چیزی در خصوص درختهای پوشا نمی‌داند تلاش کند بصورت تقریبی، رفتار الگوریتم درخت پوشا را از خود نشان بدهد! این ایده، **«هدایت بر روی مسیر معکوس»** (Reverse Path Forwarding) نامیده شده و به نظر بسیار ساده می‌رسد. وقتی یک بسته از نوع پخش فراگیر به یک مسیریاب وارد می‌شود آن مسیریاب ابتدا بررسی می‌کند که «آیا بسته از روی همان خطی وارد شده که بسته‌های معمولی برای ارسال به مبداء آن بسته، بر روی همان خط ارسال می‌شوند؟» اگر اینگونه باشد این احتمال بالا وجود دارد که بسته خودش در مسیر بهینه قرار دارد و طبعاً اولین نسخهٔ بسته‌ای است که او دریافت داشته است. در این حالت مسیریاب این بسته را بر روی تمام خطوط خود (به استثنای خطی که بسته از آن وارد شده) کپی و ارسال می‌کند. اگر بسته از روی خطی وارد شود که آن خط بر روی مسیر بهینه برای رسیدن بدان مبداء نیست بسته نادیده گرفته می‌شود چرا که احتمالاً تکراری است.

مثالی از روش «هدایت بر روی مسیر معکوس» در شکل ۵-۱۶ نشان داده شده است. بخش (الف) زیرشبکه را نشان می‌دهد و بخش (ب) درخت Sink Tree به مبداء مسیریاب I و بخش (ج) عملکرد الگوریتم «هدایت بر روی مسیر معکوس» را به تصویر کشیده است. در اولین گام، I بسته‌ای فراگیر برای F و H و J و N (یعنی گره‌های سطح دوی درخت) می‌فرستد. هر یک از این بسته‌ها از طریق خطّی دریافت می‌شوند که بر روی مسیری بهینه به I قرار دارند[1] (با فرض آنکه مسیرهای بهینه از F و H و J و N به مسیریاب I، بر روی درخت Sink Tree قرار گرفته‌اند)؛ در شکل به دور نام هر مسیریاب که بر روی مسیری بهینه به I قرار گرفته یک دایره ترسیم شده است. در گام دوم، ۸ بسته تولید می‌شود (در هر مسیریاب که در گام اول بسته‌ای را دریافت کرده، ۲ بسته تولید می‌گردد) از این هشت بسته ۵ عدد به مسیریابهایی می‌رسند که بر روی مسیر بهینه به سمت I قرار دارند [یعنی A و D و G و O و M؛ سه بستهٔ دیگر چون بر روی مسیر بهینه به I نیستند پس از دریافت می‌شوند. -م]. از شش بسته‌ای که در گام سوم تولید شده فقط سه تای آنها از طریق خطی که بر روی مسیر بهینه به I قرار دارد دریافت شده‌اند (از طریق C و E و K) و بقیه تکراری تلقی می‌شوند. پس از پنج گام و تولید ۲۴ بسته، فرآیند ارسال فراگیر به پایان می‌رسد، درحالیکه اگر ارسال بسته‌ها دقیقاً از طریق درخت Sink Tree انجام شود، تعداد گامها ۴ و کلّ بسته‌های تولیدی

---

۱. بعبارت دیگر مسیریابهای F ، H ، J ، N همگی برای ارسال یک بستهٔ معمولی برای مبداء بسته یعنی I ، از همان خطی استفاده می‌کنند که بستهٔ پخشی از روی همان خط وارد شده است. -م



شکل ۵-۱۶. «هدایت بر روی مسیر معکوس» (الف) یک زیرشبکه (ب) یک درخت Sink Tree (ج) درختی که در روش هدایت بر روی مسیر معکوس ساخته می‌شود.

۱۴ عدد خواهد بود.

مزیت اساسی روش «هدایت بر روی مسیر معکوس» آن است که هم در حد معقولی کارآمد و هم پیاده‌سازی آن ساده است. در این روش هر مسیریاب نیازی به دانستن درختهای پوشا (Spanning Tree) ندارد و سربار ناشی از قرار دادن فهرست مقاصد چندگانه در بسته‌های فراگیر تحمیل نخواهد شد. همچنین این روش نیاز به مکانیزم خاصی جهت خاتمه دادن به پروسهٔ تولید نامحدود بسته‌ها ندارد در حالی که در روش ارسال سیل‌آسا به چنین مکانیزمی نیاز است (یعنی در این روش به تمهیداتی مثل درج شمارندهٔ گام در هر بسته یا اطلاع قبلی از قطر زیرشبکه یا ذخیرهٔ فهرست بسته‌هایی که تاکنون از یک مبداء منتشر شده‌اند، نیاز نیست.)

۸-۲-۵ مسیریابی چندپخشی (Multicast Routing)

در برخی از کاربردها نیاز است که پروسه‌های مجزا و پراکنده، در یک گروه با یکدیگر کار کنند؛ به عنوان نمونه می‌توان به گروهی از پروسه‌ها که یک «سیستم پایگاه توزیع‌شدهٔ اطلاعات»[1] را پیاده‌سازی کرده‌اند، اشاره نمود. در چنین شرایطی، بطور متناوب لازم می‌شود که یک پروسه برای دیگر پروسه‌های عضو گروه خود، پیام بفرستد. اگر این گروه کوچک باشد می‌توان پیام را بطور جداگانه برای یکایک اعضاء ارسال کرد ولیکن اگر گروه بزرگ باشد این راهکار بسیار پر هزینه است. برخی اوقات می‌توان از روش پخش فراگیر (Broadcasting) بهره گرفت اما روش پخش فراگیر برای رساندن پیامی به ۱۰۰۰ ماشین بر روی شبکه‌ای که یک میلیون گره دارد، بهیچوجه کارآمد نیست زیرا اکثر گیرندگان پیام هیچ تمایلی به دریافت این گونه پیامها ندارند (بدتر از همه آنکه دیگران علاقمند به دریافت چنین پیامهایی باشند ولی مجاز به دریافت آنها نباشند!) بنابراین به راهکاری نیازمندیم که بتوان پیامهایی را برای گروه کاملاً مشخصی ارسال کرد؛ گروهی که اگرچه از لحاظ تعداد، بزرگ به نظر می‌رسد ولی در مقایسه با کلّ شبکه بسیار کوچک است.

ارسال یک پیام به یک چنین گروهی، «**چندپخشی**» (Multicasting) و به الگوریتم مسیریابی آن «**مسیریابی چندپخشی**» (Multicast Routing) گفته می‌شود. در این بخش یکی از روشهای «مسیریابی چندپخشی» را مطالعه خواهیم نمود. برای کسب آگاهی بیشتر به مراجع زیر نگاهی بیندازید:

(Chu et al., 2000; Costa et al., 2001 ; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

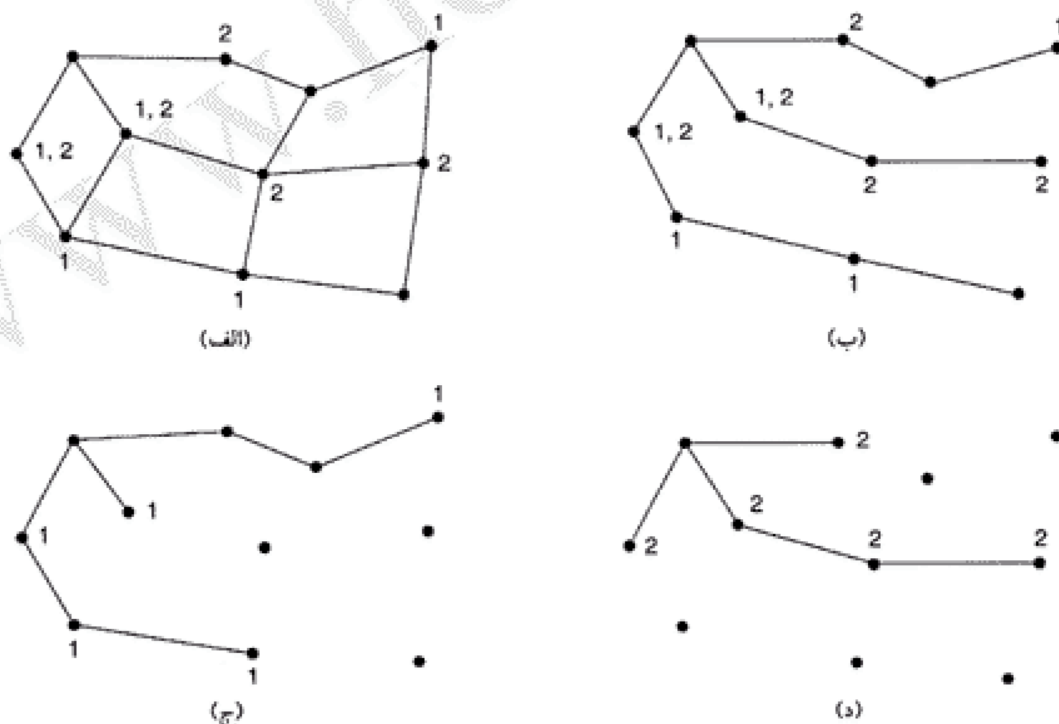
---

1. Distributed Database System

مسیریابی چندپخشی به مدیریت گروه نیاز دارد؛ یعنی به روشهایی احتیاج است که بتوان گروه ایجاد و حذف کرد و پروسه‌ها اجازه داشته باشند به گروههایی بپیوندند یا از آنها جدا شوند. چگونگی انجام چنین کارهایی به الگوریتم مسیریابی ربطی ندارد. وقتی پروسه‌ای به گروهی می‌پیوندد باید ماشین میزبان خود را از این موضوع آگاه کند. آنچه اهمیت دارد آنست که مسیریاب می‌داند هر یک از ماشینهای میزبان او عضو چه گروههایی هستند. ماشینهای میزبان باید هر گونه تغییر عضویت خود در یک گروه را به اطلاع مسیریابهای خود برسانند یا آنکه مسیریابها باید خودشان بطور متناوب در این خصوص از ماشینهای میزبان سؤال کنند. به هر حال، مسیریابها از اینکه چه ماشینی عضو چه گروهی است باخبر می‌شوند. از آن به بعد، مسیریابها به همسایه‌های خود خبر می‌دهند و بدین ترتیب این اطلاعات در کلّ زیرشبکه منتشر خواهد شد.

برای انجام عملیات مسیریابی چندپخشی، هر مسیریاب یک «درخت پوشا» (Spanning Tree) که کلّ مسیرها را در بر می‌گیرد، ایجاد می‌کند. به عنوان مثال در شکل ۵-۱۷-الف ، دو گروه ۱ و ۲ را تعریف کرده‌ایم. به نحوی که در این شکل ملاحظه می‌کنید برخی از مسیریابها به ماشینهای میزبانی متصلند که بعضاً عضو یک یا هر دوی این گروهها هستند. [در این شکل، برچسب ۱ یا ۲ بر روی هر مسیریاب، مبیّن آنست که ماشینهای میزبان متصل به آن مسیریاب عضو چه گروهی هستند. -م] در شکل ۵-۱۷-ب ، یک درخت پوشا برای مسیریاب سمت چپ نشان داده شده است.

وقتی پروسه‌ای ، یک بسته چندپخشی (Multicast Packet) را برای گروهی ارسال می‌کند اولین مسیریاب، درخت پوشای خود را بررسی کرده و آن را پیرایش (Prune) می‌کند، یعنی تمام خطوطی را که نهایتاً به ماشینهای عضو این گروه ختم نمی‌شوند، حذف می‌نماید. در این مثال شکل ۵-۱۷-ج درخت پوشا و پیرایش‌شدهٔ گروه ۱ را نشان می‌دهد. به روش مشابه شکل ۵-۱۷-د درخت پوشا و پیرایش‌شدهٔ گروه ۲ را به تصویر کشیده است. بسته‌های چندپخشی فقط از طریق درخت پوشای متناسب با گروه مقصد هدایت می‌شوند.



شکل ۵-۱۷. (الف) ساختار یک شبکه (ب) درخت پوشا برای مسیریاب سمت چپ (ج) درخت چندپخشی برای گروه ۱ (د) درخت چندپخشی برای گروه ۲.

روشهای گوناگونی برای پیرایش درخت پوشا وجود دارد. ساده‌ترین روش، زمانی قابل استفاده است که از روش «مسیریابی حالت لینک» (LS) بهره گرفته شده باشد و هر مسیریاب از توپولوژی کامل شبکه و از جمله عضویت ماشینهای میزبان در گروهها آگاه باشد. در این حالت می‌توان با شروع از انتهای هر مسیر و حرکت به سمت ریشهٔ درخت و حذف تمام مسیریابهایی که در گروه مورد نظر عضو نیستند، درخت پوشا را پیرایش کرد.

در روش «مسیریابی بردار فاصله» (DV) می‌توان از راهکار متفاوتی برای پیرایش درخت پوشا بهره گرفت. الگوریتم اصلی همان روش «هدایت بر روی مسیر معکوس» است (Reverse Path Forwarding)، ولیکن هر گاه یک مسیریاب دارای هیچ ماشینی عضو یک گروه خاص، نبوده و همچنین به هیچ مسیریاب دیگری که علاقمند به دریافت بسته‌های آن گروه است، متصل نشده باشد با ارسال یک بستهٔ خاصّ به نام PRUNE به فرستنده اعلام می‌کند که بسته‌های متعلق بدان گروه خاص را برایش نفرستد. اگر یک مسیریاب از تمام خطوط ورودیش پیغام PRUNE دریافت کند و خودش نیز ماشینی که در آن گروه عضو است نداشته باشد، او نیز پیغام PRUNE را صادر می‌نماید. در چنین حالتی، زیرشبکه به صورت بازگشتی (Recursively) پیرایش می‌شود.[1]

اشکال بالقوهٔ این الگوریتم آن است که در مقیاس شبکه‌های بزرگ، ضعیف عمل می‌کند. فرض کنید که شبکه‌ای دارای n گروه و در هر گروه بطور متوسط m عضو وجود داشته باشد. به ازای هر گروه باید m «درخت پیرایش‌شدهٔ پوشا» ایجاد و ذخیره گردد که جمعاً معادل m×n درخت است. وقتی گروههای زیادی ایجاد شده باشد برای ذخیرهٔ این درختها ، به حجم حافظهٔ قابل توجهی نیاز خواهد بود.

در طرحی دیگر، از «درختهای مبتنی بر هسته» (Core-Based Tree) بهره گرفته شده است. (Ballardie et al., 1993) در این طرح به ازای هر گروه فقط یک درخت پوشا محاسبه می‌شود، بگونه‌ای که «ریشهٔ درخت» (یا بعبارتی هسته) در نزدیکی مرکز آن گروه قرار می‌گیرد. برای ارسال یک پیام چندپخشی، ماشین میزبان آن را به سوی هسته می‌فرستد و هسته نیز عملیات پخش آن بسته را در گروه، از طریق درخت پوشا انجام می‌دهد. اگرچه این درخت برای تمام مسیریابهای مبداء کاملاً بهینه نیست ولی در عوض کاهش هزینهٔ ذخیره‌سازی m درخت به یک درخت در هر گروه، صرفه جویی بسیار قابل توجهی محسوب می‌شود.

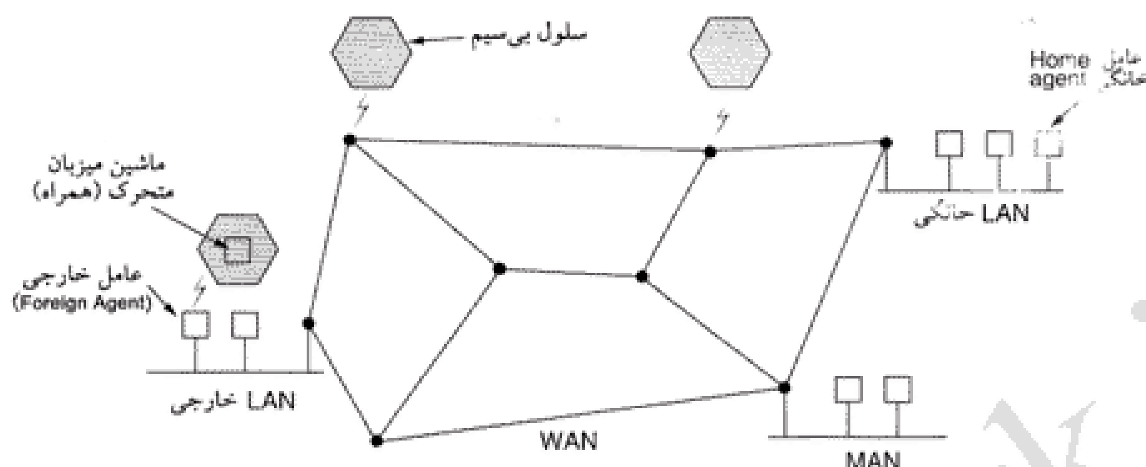
۵-۲-۹ مسیریابی برای ماشینهای متحرک

امروزه میلیونها نفر از مردم کامپیوترهای قابل حمل دارند و عموماً تمایل دارند در هر کجای دنیا که باشند نامه‌های الکترونیکی خود را بخوانند و به سیستم فایل همیشگی خود دسترسی داشته باشند. این ماشینهای متحرک مشکل جدیدی را بوجود می‌آورند: قبل از هدایت یک بسته به سوی ماشین متحرک، در ابتدا شبکه باید آن ماشین را پیدا کند. موضوع پیوستن ماشینهای میزبان متحرک (Mobile Hosts) به شبکه، موضوعی بسیار جدید و نو است ولیکن در این بخش به ارائهٔ برخی از راهکارها در این زمینه خواهیم پرداخت.

مدلی که طراحان شبکه عموماً برای ساختار ارتباطی جهان در نظر می‌گیرند در شکل ۵-۱۸ نشان داده شده است. در این شکل یک WAN با تعدادی مسیریاب و ماشین میزبان دیده می‌شود. به WAN تعدادی LAN ،

---

۱. بعبارت ساده‌تر در ابتدای کار هر بستهٔ چندپخشی بروش «هدایت بر روی مسیر معکوس» که در بخش قبل بررسی شد بر روی خطوط خروجی مسیریابها هدایت می‌شوند و تا اینجای کار هیچ تفاوتی با پخش فراگیر ندارد. به محض پخش اولین بسته بصورت فراگیر، تمام مسیریابهایی که انتظار دریافت چنین بسته‌ایی را نداشته‌اند با ارسال بستهٔ کنترلی Prune از همسایهٔ خود که این بسته را برایشان فرستاده خواهش می‌کنند این کار را تکرار نکند. وقتی یک مسیریاب خودش هیچ عضوی در این گروه ندارد و تمام همسایه‌هایش نیز با ارسال Prune از او خواسته‌اند که بسته‌های آن گروه را برایشان نفرستد، لزومی به دریافت بسته‌های آن گروه نمی‌بیند و او هم پیغام Prune به همسایهٔ قبلی خود می‌فرستد. بدین ترتیب از آخر مسیر به اول درخت پوشای فرضی اصلاح می‌شود. -م



شکل ۵-۱۸. یک شبکهٔ WAN که شبکه‌های LAN، MAN و سلولهای بی‌سیم بدان متصل شده‌اند.

MAN و شبکهٔ بی‌سیم (از نوعی که در فصل ۲ مطالعه کردیم)، متصل شده است.

به ماشینهای میزبان که حرکت نمی‌کنند ماشینهای ثابت (Stationary) گفته می‌شود. این ماشینها از طریق سیمهای مسی یا فیبرهای نوری به شبکه متصل شده‌اند. برعکس، دو نوع دیگر ماشین میزبان، قابل رؤیت است: «ماشینهای میزبان مهاجر» (Migratory Hosts) ماشینهای ثابتی هستند که گاه‌به‌گاه از یک سایت ثابت به سایتی دیگر نقل مکان می‌کنند ولی فقط زمانی در شبکه قرار می‌گیرند که اتصال فیزیکی آنها برقرار شود. «ماشینهای میزبان سیّار» (Roaming Hosts) در حال حرکت هم کار می‌کنند و باید ارتباط خود را در حین حرکت حفظ کنند. ما این دو رده از ماشینهای میزبان را «ماشینهای میزبان متحرک» (Mobile Hosts) می‌نامیم: یعنی تمام ماشینها دور از محل استقرار همیشگی خود هستند و می‌خواهند به شبکه متصل شوند.

فرض بر آن است که تمام ماشینها دارای یک **«محل استقرار دائمی»** (Home Location) هستند که هیچگاه تغییر نمی‌کند. همچنین هر ماشین میزبان دارای آدرس ثابتی در محل استقرار دائم خود است که موقعیت این محل را مشخص می‌کند (شبیه به یک شماره تلفن در آمریکا مثل ۱-۲۱۲-۵۵۵۱۲۱۲، با کد کشوری ۱ و کد منهتن، ۲۱۲). هدف نهایی فرآیند مسیریابی در سیستمی با ماشینهای متحرک، آنست که بتوان بسته‌ها را به کمک آدرس دائم آنها به ماشینها رساند و ماشینهای میزبان (در هر کجا که باشند) بتوانند بسته‌های خود را تحویل بگیرند. ظریفترین بخش قضیه آنست که ابتدا باید این ماشینها را پیدا کرد.

در مدل شکل ۵-۱۸ کل دنیا از نظر جغرافیایی به چندین بخش کوچک تقسیم شده است. اجازه بدهید آنها را «ناحیه» بنامیم؛ «ناحیه»، خود یک شبکهٔ LAN یا یک «سلول بی‌سیم» (Wireless Cell) است. در هر ناحیه یک یا چند **«عامل خارجی»** (Foreign Agent) وجود دارد که در عمل پروسه‌هایی هستند که فهرست ماشینهای متحرک و میهمان آن ناحیه را پیگیری و مدیریت می‌کنند. بعلاوه هر ناحیه یک **«عامل خانگی»** (Home Agent) دارد که فهرست ماشینهایی را نگهداری می‌کند که محل استقرار دائمی آنها همین ناحیه است ولی فعلاً در نواحی دیگر به سر می‌برند.

وقتی ماشین جدیدی وارد یک ناحیه می‌شود (از طریق اتصال فیزیکی با شبکه مثلاً با اتصال کابل آن به LAN و یا با پرسه زدن در درون سلول بی‌سیم)، باید خودش را در «عامل خارجی» ثبت‌نام نماید. روال ثبت نام عموماً به شکل زیر انجام می‌شود:

۱. هر **«عامل خارجی»** بطور متناوب یک بستهٔ فراگیر پخش کرده و حضور خود و آدرسش را به اطلاع همه

می‌رساند. یک ماشین متحرک تازه وارد باید منتظر چنین پیامهایی بماند ولیکن اگر در اسرع وقت، چنین پیامی نرسد ماشین متحرک می‌تواند بسته‌ای فراگیر برای همه بفرستد با این مضمون که: «آیا هیچ عامل خارجی در اینجا هست؟!»

۲. ماشین متحرک در عامل خارجی ثبت‌نام می‌کند، آدرس محل استقرار دائم خود، آدرس فیزیکی (یعنی آدرس لایهٔ پیوند داده‌ها) فعلی خود و برخی اطلاعات امنیتی (جهت احراز هویت) را ارائه می‌دهد.

۳. عامل خارجی با عامل خانگی آن ماشین متحرک تماس گرفته و می‌گوید: یکی از ماشینهای شما در ناحیهٔ ما به سر می‌برد! در این پیام که از سوی «عامل خارجی» به سوی «عامل خانگی» ارسال می‌شود آدرس شبکهٔ عامل خارجی [آدرس خودش] مشخص می‌شود. این پیام همچنین شامل برخی اطلاعات امنیتی است تا عامل خانگی را متقاعد کند که ماشین متحرک واقعاً در ناحیهٔ اوست.

۴. عامل خانگی اطلاعات امنیتی ارائه شده را که محتوی «مهر زمان» (Timestamp) نیز هست بررسی می‌کند تا برایش ثابت شود که این پیام در خلال همین چند ثانیهٔ اخیر تولید شده است. اگر متقاعد شود به «عامل خارجی» اجازهٔ ادامهٔ کار را می‌دهد.

۵. وقتی «عامل خارجی» پیام تأیید «عامل داخلی» را دریافت کند یک درایه (Entry) در جدول خود ایجاد کرده و به ماشین متحرک اعلام می‌کند که ثبت‌نام شده است.

آرمانی آن است که وقتی یک ماشین میزبان ناحیه‌ای را ترک می‌کند به همان ترتیب اعلام کند تا عضویت آن لغو شود، ولی بسیاری از کاربران به محض آنکه کارشان تمام می‌شود بلافاصله کامپیوتر خود را خاموش می‌کنند و مهلتی برای اعلام ترک ناحیه، باقی نمی‌ماند!

وقتی بسته‌ای برای یک ماشین متحرک ارسال می‌شود ابتدا آن بسته به سوی شبکهٔ LAN خانگی او [یعنی محل استقرار دائم او] مسیریابی و هدایت می‌شود زیرا در حقیقت بسته به آدرس دائم او ارسال شده است. (مرحلهٔ ۱ از شکل ۵-۱۹) در این شکل فرستنده‌ای در شهری از سیاتل آمریکا می‌خواهد بسته‌ای را برای یک ماشین میزبان متحرک در نیویورک بفرستد. بسته ابتدا به آدرس شبکهٔ LAN خانگی او در نیویورک ارسال می‌شود و در آنجا توسط «عامل خانگی» تحویل گرفته می‌شود. عامل خانگی در حافظهٔ خود محل استقرار جدید (و موقت) ماشین متحرک را جستجو کرده و آدرس عامل خارجی که آن ماشین را در پوشش خود گرفته (مثلاً در لوس آنجلس) پیدا می‌کند.

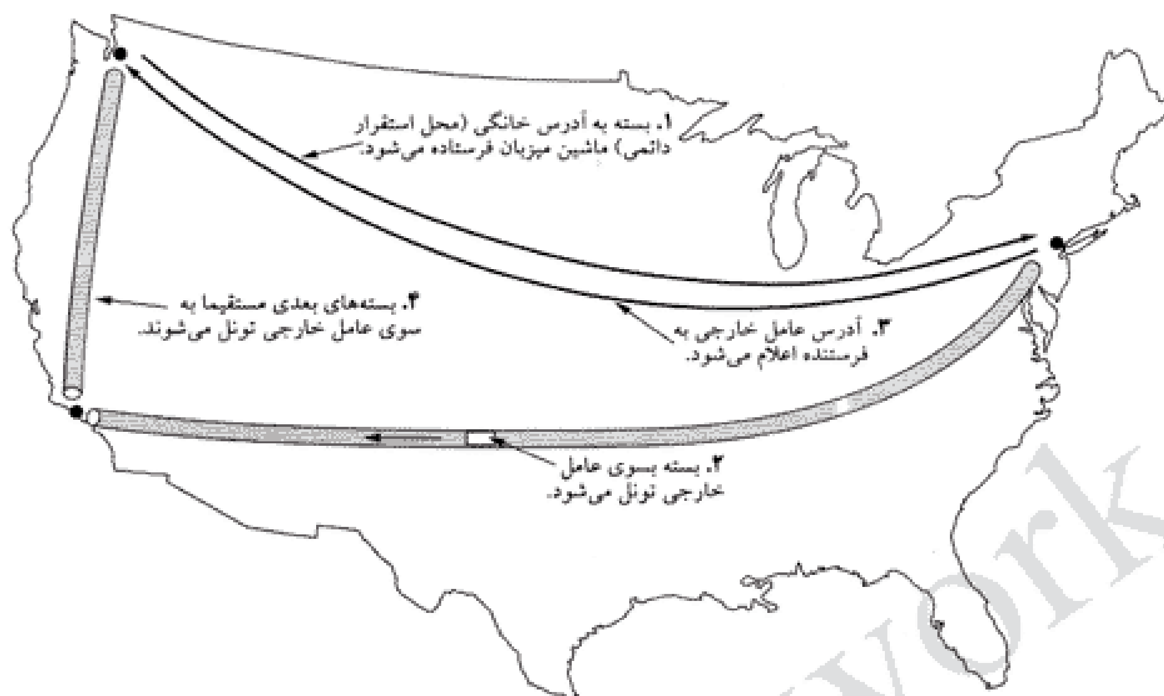
در این لحظه، عامل خانگی دو کار انجام می‌دهد: اول آن که این بسته را در درون فیلد داده از یک بستهٔ بیرونی دیگر جاسازی کرده و آن را برای عامل خارجی می‌فرستد. (مرحلهٔ ۲ از شکل ۵-۱۹) به این مکانیزم «ایجاد تونل» (Tunneling) گفته می‌شود و بعداً به تفصیل در مورد آن صحبت خواهیم کرد. پس از دریافت بستهٔ جاسازی‌شده توسط عامل خارجی، بستهٔ اصلی از درون فیلد دادهٔ آن جدا شده و به ماشین متحرک تحویل می‌شود.

دوم آنکه عامل خانگی به فرستندهٔ بسته اعلام می‌کند که از این به بعد بسته‌های خود را با جاسازی درون یک بستهٔ دیگر (که به صراحت آدرس «عامل خارجی» در آن درج شده) مستقیماً به عامل خارجی بفرستد. (مرحلهٔ ۳) از آن به بعد بدون در نظر گرفتن موقعیت دائمی ماشین متحرک، بسته‌ها مستقیماً به سوی عامل خارجی هدایت شده و تحویل ماشین متحرک می‌شود.[1]

روشهایی که تاکنون معرفی شده‌اند از چندین جهت با هم تفاوت دارند: اولین مورد آن که چه مقدار از وظایف

---

۱. در فرآیند تونل یک بستهٔ کامل در درون یک بستهٔ دیگر جاسازی می‌شود: آدرس بستهٔ بیرونی مربوط به «عامل خارجی» است و آدرس بستهٔ درونی هویت صاحب اصلی بسته یعنی ماشین متحرک را مشخص می‌کند. -م



شکل ۵-۱۹. مسیریابی بسته‌ها بسوی ماشینهای متحرک.

این پروتکل توسط مسیریابها و چه مقدار توسط ماشینهای میزبان انجام می‌شود و آن بخش از وظایف که در ماشین میزبان باید انجام شود در چه لایه‌ای تعریف شده است. دوم آن که در برخی از روشها، مسیریابهای واقع بر روی مسیر قادرند «نگاشتهای آدرس» [یعنی تغییر آدرس ماشین متحرک به آدرس جدید] را ثبت کرده و بدون هیچ دخالت بیرونی در میانهٔ راه جلوی ترافیک ارسالی به آدرس قبلی را گرفته و آن را به سمت آدرس جدید تغییر مسیر بدهند.

سوم آن که در برخی دیگر از این روشها به هر ماشین متحرک که میهمان شبکه‌ای جدید شده یک آدرس موقت ولیکن منحصر به فرد داده می‌شود در حالی که در برخی دیگر این آدرس موقت مربوط به آن عامل خارجی می‌باشد که هدایت ترافیک تمام میهمانان خود را بر عهده گرفته است.

تفاوت چهارم این روشها آنست که وقتی بسته‌ها برای رسیدن به یک ماشین دیگر آدرس‌دهی شده‌اند چگونه تحویل ماشین دیگری می‌شوند. [بدین معنا که بسته‌هایی که به سوی آدرس قبلی یک ماشین روانه شده‌اند به چه نحو برای هدایت به جایی دیگر تغییر آدرس می‌دهند. -م] گزینهٔ اول آنست که در هر بسته، آدرس مقصد عوض شود و بستهٔ تغییر یافته از نو ارسال شود. گزینهٔ دوم آنست که کل بسته به همراه آدرس دائم (خانگی) و با کلیهٔ مشخصات [به صورت دست نخورده]، در درون فیلد داده از یک بستهٔ دیگر جاسازی (کپسوله) شود و بستهٔ جدید به آدرس موقت ماشین ارسال گردد.

تفاوت آخر آنکه روشهای مختلف از دیدگاه تمهیدات امنیتی با یکدیگر فرق می‌کنند. عموماً وقتی یک ماشین میزبان یا مسیریاب، پیامی با مضمون این مثال دریافت می‌کند: «لطفاً از هم اکنون به بعد تمام نامه‌های استفانی را برای من بفرست!» دو سؤال مطرح می‌شود: این پیام واقعاً متعلق به کیست (با چه کسی صحبت می‌شود) و آیا چنین کاری اصولاً به صلاح است؛ در مراجع زیر چندین پروتکل در خصوص ماشینهای متحرک تشریح و مقایسه شده‌اند:

(Hac and Guo, 2000; Perkins, 1998a; Snoeren and Balakrishnan, 2000; Solomon, 1998; Wang and Chen, 2001)

۱۰-۲-۵ مسیریابی در شبکه‌های ویژه (Routing in Ad Hoc Networks)

تاکنون روشهای مسیریابی در محیطهایی را که ماشینهای میزبان متحرکند ولی مسیریابها ثابت هستند، بررسی کرده‌ایم. یک حالت استثنایی آن است که مسیریابها نیز خودشان متحرک باشند! از چنین محیطهایی می‌توان به موارد زیر اشاره کرد:

۱. وسایل نقلیهٔ نظامی در صحنهٔ نبرد بدون دسترسی به هیچگونه زیرساخت ارتباطی
۲. ناوگان دریایی بر روی دریا
۳. نیروی امداد در شرایط اضطراری مثل زلزله که کلیهٔ زیرساختها را نابود کرده است
۴. گردهمایی افراد با کامپیوتر کیفی، در ناحیه‌ای بدون شبکهٔ بی‌سیم 802.11

در تمام این موارد (و نظایر آن) هر گره شامل یک مسیریاب و یک ماشین میزبان است که اغلب هر دوی آنها بر روی یک ماشین قرار می‌گیرند. شبکه‌ای از این گره‌ها که در کنار هم قرار می‌گیرند اصطلاحاً **«شبکه‌های ویژه»** یا MANET[۱] نامیده می‌شود. اجازه بدهید مختصراً این شبکه‌ها را بررسی نماییم. برای آگاهی بیشتر می‌توان به (Perkins, 2001) مراجعه کرد.

آنچه که «شبکه‌های ویژه» را از «شبکه‌های سیمی» متمایز می‌کند آن است که تمام قواعد طبیعی در خصوص توپولوژی ثابت، همسایه‌های شناخته شدهٔ ثابت، تناظر دائم بین موقعیت فیزیکی و آدرس IP ماشینها و مواردی از این قبیل در خصوص «شبکه‌های ویژه» صادق نیست. مسیریابها در رفت و آمد هستند و ممکن است در هر لحظه از محل جدیدی سر در آورند! در یک شبکهٔ سیمی هر گاه یک مسیریاب، مسیری معتبر به برخی از نقاط مقصد در شبکه داشته باشد آن مسیر تا ابد معتبر خواهد بود (مگر آن که در جایی از سیستم خرابی بوجود بیاید). در یک شبکهٔ ویژه، توپولوژی شبکه به طور دائم در تغییر است لذا اعتبار مسیرها و بهینگی آنها به ناگاه و بی‌هیچ هشدار قبلی تغییر می‌کند. آشکار است که با چنین وضعیتی، مسیریابی در شبکه‌های ویژه کاملاً متفاوت از شبکه‌های ثابت می‌باشد.

گونه‌های متعددی از الگوریتمهای مسیریابی برای شبکه‌های ویژه پیشنهاد شده است. یکی از جالبترین آنها الگوریتم مسیریابی AODV[۲]) است.(Perkins and Royer, 1999) این الگوریتم گونه‌ای از «الگوریتم بردار فاصلهٔ بلمن-فورد» محسوب می‌شود که برای کار در محیطهای متحرک تطبیق داده شده و در آن پهنای باند محدود و عمر کم باطری ماشینها در این محیط [در محاسبات مربوط به مسیرهای بهینه] در نظر گرفته شده است. یکی دیگر از ویژگیهای نامتعارف این روش آن است که الگوریتم «برحسب تقاضا» (On-Demand) عمل می‌کند بدین معنا که مسیر رسیدن به برخی از نقاط مقصد، فقط وقتی تعیین می‌شود که کسی بخواهد بسته‌ای را بدان مقصد بفرستد. اجازه بدهید ببینیم قضیه از چه قرار است.

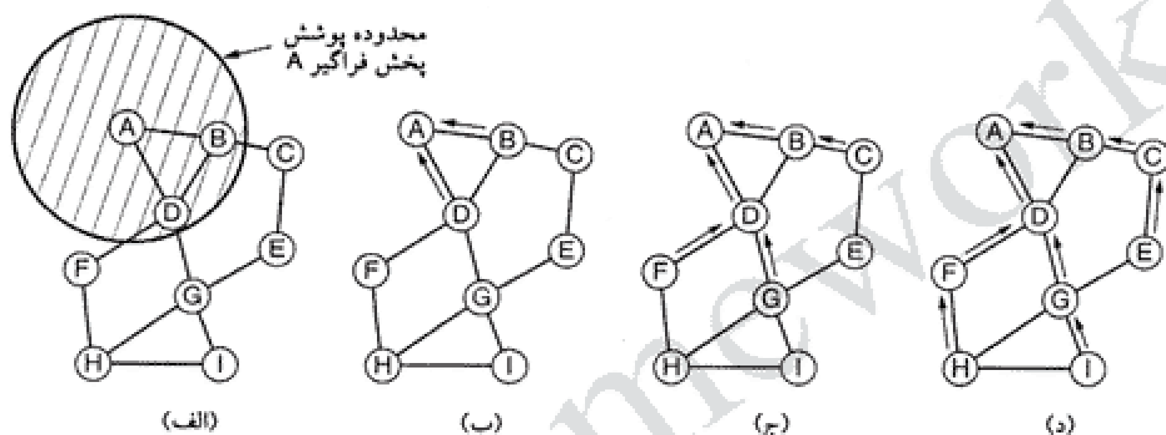
کشف مسیر

یک شبکهٔ ویژه را در هر لحظه از زمان می‌توان با گرافی از گره‌ها توصیف کرد. (گره‌ها عبارتند از مسیریابها + ماشینهای میزبان) اگر دو ماشین بتوانند از طریق سیستم رادیویی خود مستقیماً با یکدیگر ارتباط برقرار کنند می‌گوییم این دو گره (Node) بهم متصل هستند. (یعنی در گراف شبکه، بین این دو گره یک کمان وجود دارد). از آنجایی که امکان دارد یکی از این دو ماشین، فرستندهٔ پرقدرت‌تری نسبت به دیگری داشته باشد لذا این امکان وجود دارد که ارتباط A به B برقرار باشد ولی ارتباط B با A میسّر نباشد؛ ولیکن برای سادگی فرض را بر آن می‌گذاریم که تمام ارتباطات، دو طرفه و متقارن هستند. همچنین باید بدین نکته اشاره کرد که هر گاه دو گره در برد

۱. Mobile Ad Hoc Network
۲. Ad Hoc On-Demand Distance Vector

رادیویی یکدیگر باشند تضمینی وجود ندارد که ارتباط آنها برقرار باشد؛ ممکن است بین آنها ساختمان، تپه یا موانع دیگری وجود داشته باشد که جلوی ارتباط آنها را بگیرد. [هر چند در برد یکدیگر واقعند.]

برای توصیف الگوریتم، شبکهٔ ویژهٔ شکل ۵-۲۰ را در نظر بگیرید که در آن یک پروسه در گره A می‌خواهد بسته‌ای را برای گره I بفرستد. در الگوریتم AODV ، هر گره دارای جدولی است که کلید این جدول، آدرس مقصد است[1] و هر یک از رکوردهای این جدول، اطلاعاتی در خصوص مقصد و آنکه برای رساندن بسته‌ای به آن مقصد باید بسته را به کدامیک از همسایه‌های آن فرستاد، در خود نگاهداری می‌کنند. فرض کنید که A در جدول خود جستجو کرده و هیچ درایه‌ای متناظر با I در آن نمی‌یابد. حال بایستی مسیری به I کشف کند. همین ویژگی که مسیرها فقط در هنگام لزوم کشف می‌شوند به الگوریتم، ویژگی On-Demand یعنی «برحسب تقاضا» داده است.



شکل ۵-۲۰. (الف) محدودهٔ پوشش پخش فراگیر A (ب) پس از آنکه B و D پخش فراگیر A را دریافت کردند. (ج) پس از آنکه C و F و G پخش فراگیر A را دریافت کردند. (د) پس از آنکه E و H و I پخش فراگیر A را دریافت کردند. گره‌های سایه‌دار دریافت کنندگان جدید هر مرحله محسوب می‌شوند. فلشها مسیر معکوس (مسیر برگشت) را مشخص می‌کنند.

برای پیدا کردن موقعیت I ، گره A یک بستهٔ خاص به نام ROUTE REQUEST (تقاضای مسیر) ساخته و آن را به صورت فراگیر منتشر می‌کند. به‌گونه‌ای که در شکل ۵-۲۰-الف مشهود است، این بسته به B و D می‌رسد. در حقیقت دلیل آنکه در این گراف ، B و D به A متصل شده‌اند آنست که قادرند سیگنال مخابراتی A را دریافت کنند. به عنوان مثال بین گره F و A در گراف هیچ کمانی ترسیم نشده است چراکه F نمی‌تواند سیگنال رادیویی A را دریافت کند. بنابراین ارتباط مستقیمی بین F و A وجود ندارد.

قالب بستهٔ ROUTE REQUEST در شکل ۵-۲۱ نشان داده شده است. این بسته شامل «آدرس مبداء» و «آدرس مقصد» است و مشخص می‌کند که چه کسی در جستجوی چه کسی است. (عموماً از آدرسهای IP آنها استفاده می‌شود) این بسته همچنین حاوی یک **«شناسهٔ تقاضا»** (Request ID) است. این شناسه در حقیقت یک شمارندهٔ محلی است که در هر گره وجود دارد و هرگاه که یک بستهٔ ROUTE REQEST منتشر گردید یک واحد به آن اضافه می‌شود. ترکیب «آدرس مبداء» و فیلد «شناسهٔ تقاضا» ، هویت بستهٔ ROUTE REQUEST را به صورت یکتا و منحصر بفرد تعیین کرده و بدین ترتیب گره‌ها قادرند بسته‌هایی که احتمالاً به صورت تکراری دریافت می‌شوند را تشخیص داده و حذف کنند.

هر گره به غیر از «شمارندهٔ شناسهٔ تقاضا» یک شمارندهٔ دیگر هم دارد که هرگاه بستهٔ ROUTE REQUEST

۱. یعنی جستجوی رکوردها در این جدول براساس آدرس مقصد انجام می‌شود. ــم

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

شکل ۵-۲۱. قالب بستهٔ ROUTE REQUEST.

ارسال شود (یا پاسخی برای بستهٔ ROUTE REQUEST دیگران دریافت گردد) یک واحد بدان اضافه می‌گردد. عملکرد این شمارنده تا حدودی شبیه به یک ساعت است و کاربرد آن تشخیص مسیر جدید از مسیر قدیمی است. [یعنی وقتی دو بستهٔ یکسان در مورد مشخصات یک مسیر دریافت می‌شود این شمارنده مشخص می‌کند که کدام جدیدتر از دیگری است] فیلد چهارم از شکل ۵-۲۱ مقدار همین شمارنده را برای گره مبداء (که در این مثال A است) مشخص می‌کند. [در این مثال A در جستجوی I است لذا در بستهٔ ROUTE REQUEST شمارهٔ ترتیب تقاضای خود را درج کرده است.] فیلد پنجم مشخص کنندهٔ آخرین و جدیدترین شمارهٔ بسته‌ای است که A در مورد I دریافت کرده است. (و صفر است اگر تاکنون در مورد I چیزی دریافت نکرده باشد.) آخرین فیلد یعنی **«شمارندهٔ گام»** (Hop Counter) مشخص می‌کند که بسته تاکنون چند گام را طی کرده است. مقدار اولیهٔ این فیلد صفر است و به ازای عبور از هر گره یک واحد به آن اضافه می‌شود.

وقتی بستهٔ ROUTE REQUEST به یک گره می‌رسد (در این مثال به B و D) طبق مراحل زیر پردازش می‌شود:

۱. ابتدا جفت مشخصهٔ (آدرس مبداء، شناسهٔ تقاضا) در یک جدول محلی (که سوابق دریافت چنین بسته‌هایی را نگاهداری می‌کند) جستجو می‌شود تا مشخص گردد که آیا این بسته قبلاً نیز دریافت و پردازش شده است؟ اگر تکراری بود بسته حذف شده و پردازش آن در همین جا خاتمه می‌یابد. اگر تکراری نبود این زوج مشخصهٔ بسته را در جدول سوابق وارد می‌کند تا در آینده نیز بسته‌ای مشابه با این بسته پردازش نشود. سپس فرآیند پردازش ادامه می‌یابد.

۲. سپس گیرنده در جدول مسیریابی خود، آدرس مقصد را جستجو می‌کند. اگر یک مسیر جدید و **«تازه»** به این مقصد پیدا شود یک بستهٔ ROUTE RERLY برای مبداء برگردانده می‌شود تا مبداء نیز از چگونگی رسیدن به مقصد مورد نظر آگاه شود. (اغلب، پاسخ بدین نحو است: از طریق من عمل کن!) «تازه» بودن مسیر بدین معنی است که فیلد «شمارهٔ ترتیب مقصد» ذخیره شده در جدول مسیریابی باید بزرگتر یا مساوی همین فیلد در بستهٔ ROUTE REQUEST باشد. اگر کمتر بود بدین معنی تلقی می‌شود که مشخصات ذخیره شده در حافظه، قدیمی‌تر از مسیری است که خود مبداء برای رسیدن به مقصد در اختیار دارد[1]، در این صورت مرحلهٔ سوم به اجرا در می‌آید.

۳. از آنجایی که گیرنده هیچ مسیر جدیدی بدان مقصد نمی‌شناسد، به مقدار فیلد «شمارندهٔ گام» (Hop Count) یک واحد اضافه کرده و بستهٔ ROUTE REQUEST را مجدداً پیرامون خود منتشر می‌نماید. البته داده‌های درون بسته را استخراج کرده و آن را به عنوان یک درایهٔ جدید در جدول **«مسیرهای معکوس»** ذخیره می‌کند.[2] این اطلاعات بدان جهت مفید است که می‌توان مسیرهای معکوس ایجاد کرد و از

---

۱. کاربرد فیلد شمارهٔ ترتیب مقصد (Destination Sequence #) را در ذهن خود اینگونه فرض کنید که مثلاً A به همسایه‌های خود اعلام می‌کند که من خودم اطلاعاتی در خصوص مسیر رسیدن به I دارم که شمارهٔ ترتیب آن (مثلاً) Destination Sequence #=۱۲۳۴ است. اگر شما اطلاعات جدیدتری دارید لطفاً برای من بفرستید! -م

۲. یعنی اگرچه مبداء به دنبال یافتن یک مقصد خاصّ است ولیکن در بستهٔ تقاضا حداقل خود را معرفی کرده است فلذا مسیریابهای گیرندهٔ این تقاضا، لااقل می‌توانند از حضور این مبداء و مسیر رسیدن به آن کسب آگاهی کنند؛ به چنین مسیری «مسیر معکوس» گفته می‌شود. -م

طریق آن، در آینده پاسخ این تقاضا را به مبداء برگرداند. فلشهایی که بر روی شکل ۵-۲۰ دیده می‌شوند چگونگی ساخته شدن مسیرهای معکوس را نشان می‌دهند. به محض ایجاد یک مسیر معکوس و جدید، یک زمان‌سنج (تایمر) برای آن تنظیم می‌شود. اگر مهلت این زمان‌سنج منقضی شود و پاسخی به بستهٔ ROUTE REQUEST برنگردد، مسیر ایجاد شده حذف خواهد شد.

هیچیک از گره‌های B و D نمی‌دانند که I کجاست لذا این دو نیز ضمن ساختن مسیر معکوس جهت بازگشت پاسخ به A، با تغییر فیلد Hop Count به ۱، آنرا مجدداً منتشر می‌کنند.(به شکل ۵-۲۰ دقت کنید.) انتشار مجدّد بسته توسط B، به C و D می‌رسد. C نیز درایه‌ای در جدول مسیرهای معکوس خود ایجاد و آن را از نو منتشر می‌کند. در مقابل، D آنرا به عنوان بسته‌ای تکراری حذف می‌کند. [چون قبلاً از طریق A دریافت کرده است.] به همین طریق بستهٔ منتشره توسط D در B تکراری تشخیص داده شده و حذف می‌گردد. ولیکن به گونه‌ای که در شکل ۵-۲۰-ج دیده می‌شود بستهٔ منتشره توسط D در F و G پذیرفته و ذخیره می‌شود.
پس از آن که E و H و I نیز بستهٔ منتشر شده را دریافت کردند، عاقبت پیغام ROUTE REQUEST به مقصد خود می‌رسد (یعنی به خود I یا به گره‌ای که از موقعیت I خبر دقیق دارد می‌رسد). (شکل ۵-۲۰-د را ببینید.) اگرچه ما کلّ فرآیند انتشار را در سه مرحلهٔ جدا نشان داده‌ایم ولیکن انتشار بسته‌ها از گره‌های مختلف به صورت هماهنگ و همزمان انجام نمی‌شود.

گره I در پاسخ به تقاضای ورودی، یک بستهٔ ROUTE REPLY مطابق با شکل ۵-۲۲ ایجاد می‌کند. فیلدهای **«آدرس مبداء»**، **«آدرس مقصد»** مستقیماً از بستهٔ تقاضا استخراج و در بستهٔ پاسخ کپی می‌شوند ولیکن **«شمارهٔ ترتیب مقصد»** (Destination Sequence Number) برگرفته از مقدار شمارنده‌ای است که درون حافظهٔ I قرار دارد. فیلد «شمارندهٔ گام» نیز به صفر تنظیم می‌شود. فیلد «طول عمر» (Lifetime) مشخص می‌کند که مشخصات مسیر اعلام شده تا چه زمانی معتبر است. این بسته صرفاً برای گره‌ای ارسال می‌شود که بستهٔ تقاضا (یعنی ROUTE REQUEST) از طریق او دریافت شده است (در این مثال گره G). این بسته، مسیر معکوس خود به D و نهایتاً به A را طی می‌کند. در هر گره به مقدار فیلد شمارندهٔ پیام یک واحد اضافه می‌شود تا هر گره که آن را می‌بیند بفهمد که فاصله‌اش تا گره مقصد (در این مثال I) چقدر است.

Source address	Destination address	Destination sequence #	Hop count	Lifetime

شکل ۵-۲۲. قالب بستهٔ ROUTE REPLY.

هر گره میانی این بسته را در راه بازگشت، بررسی می‌کند. اگر یکی از سه شرط زیر برقرار باشد، اطلاعاتی در خصوص مسیر رسیدن به I، در جدول مسیریابی هر گره میانی ذخیره خواهد شد:

۱. اگر هیچ مسیر شناخته شده‌ای به I نداشته باشد.
۲. اگر شمارهٔ ترتیب I [یعنی شماره‌ای که I در فیلد Destination Seq.No. گذاشته است] بزرگتر از شماره‌ای باشد که در جدول مسیریابی درج شده است.
۳. اگر شماره ترتیب یکسان باشد ولی مسیر جدید کوتاهتر باشد. [کوتاه بودن از فیلد شمارندهٔ گام مشخص می‌شود.]

در این روش تمام گره‌هایی که بر روی مسیر معکوس قرار دارند، مجاناً از مسیر رسیدن به I آگاه می‌شوند (یکی از محاسن جانبی کشف مسیر توسط A، آگاهی گره‌های میانی است). گره‌هایی که بستهٔ ROUTE REQUEST را

در مسیر رفت دریافت کرده‌اند ولیکن در مسیر معکوس نیستند (در این مثال B و C و E و F و H) پس از منقضی شدن زمان‌سنج (تایمر)، مسیر معکوس به A را حذف می‌کنند.

در شبکه‌های عظیم، این الگوریتم بسته‌های فراگیر بسیار زیادی را تولید می‌کند، حتی اگر گره مقصد در نزدیکی مبداء باشد. برای کاهش تولید بسته‌ها در فرآیند انتشار، می‌توان بدین نحو عمل کرد: فیلد TTL (Time To Live) در بستهٔ IP، توسط فرستندهٔ آن به مقداری نزدیک به «قطر»[1] شبکه تنظیم شده و به ازای عبور از هر گره، یک واحد از آن کسر گردد؛ هر گاه به صفر رسید، بسته به جای انتشار مجدد حذف شود.

فرآیند کشف مسیر را می‌توان بدین نحو اصلاح کرد: برای یافتن موقعیت مقصد، فرستنده، بستهٔ ROUTE REQUEST را با تنظیم فیلد TTL به ۱ ارسال می‌کند. اگر در یک مدت زمان معقول پاسخی برنگشت بستهٔ بعدی را با TTL معادل ۲ ارسال می‌نماید. همین روال با مقادیر ۳ و ۴ و ۵ و... ادامه می‌یابد تا بالاخره پاسخی برگردد. در این روش، جستجو ابتدا به صورت محلی و در پیرامون گره شروع شده و در هر مرحله محدودهٔ جستجو گسترده‌تر می‌شود.

نگهداری مسیر (Route Maintenance)

از آنجایی که گره‌ها می‌توانند جابجا شده یا کلاً خاموش شوند لذا توپولوژی شبکه گاه‌به‌گاه تغییر می‌کند. به عنوان مثال در شکل ۵-۲۰ اگر G به ناگاه خاموش شود، A نخواهد فهمید مسیری که برای رسیدن به I در اختیار داشته (یعنی مسیر ADGI) دیگر برقرار نیست. این الگوریتم باید بتواند به نحوی این مسئله را حل و فصل کند.
هر گره در شبکه بطور متناوب «پیغام سلام» (Hello Message) منتشر می‌کند. انتظار می‌رود که همسایه‌ها به این پیام پاسخ بدهند. اگر پاسخی باز نگشت، منتشرکنندهٔ پیام آگاه می‌شود که همسایهٔ او از بُردش خارج شده و دیگر ارتباط آنها برقرار نیست. به روش مشابه اگر بسته‌ای معمولی برای همسایهٔ خود بفرستد و پاسخی نگیرد متوجه می‌شود که آن همسایه در دسترس نیست.

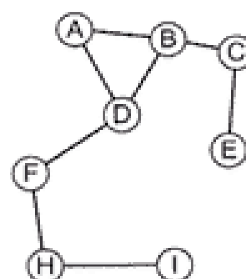
از این اطلاعات می‌توان برای پاک کردن مسیرهایی که دیگر کار نمی‌کنند بهره گرفت. هر گره مثل N برای یکایک گره‌های مقصد که همسایه‌هایش در خلال ΔT ثانیهٔ گذشته بسته‌ای را از طریق او بدان مقصد ارسال کرده‌اند، فهرستی را نگه می‌دارد. این فهرستها اصطلاحاً **«همسایه‌های فعال N»** نامیده می‌شوند. بدین منظور گره N دارای جدولی است که کلید آن، آدرس مقصد گره‌های شبکه است؛ همچنین در این فهرست، گره بعدی برای رسیدن بدان مقصد، تعداد گام برای رسیدن به آن مقصد، «آخرین شمارهٔ ترتیب» و «فهرست همسایه‌های فعال» درج شده است. به عنوان نمونه جدول مسیریابی گره D برای توپولوژی مثال قبلی، چیزی شبیه به شکل ۵-۲۳-الف است.

وقتی یکی از همسایه‌های N از دسترس خارج می‌شود، گره N جدول مسیریابی خود را بررسی می‌کند تا ببیند کدامیک از گره‌های شبکه در مسیرهای خود از گره حذف شده استفاده می‌کرده‌اند. این موضوع به همسایه‌های فعال اطلاع داده می‌شود که: تمام مسیرهای آنها از طریق N نامعتبر است (چرا که N بسته‌های آنها از طریق گره حذف شده ارسال می‌شده است)؛ همسایه‌های فعال نیز به همسایه‌های فعال خود خبر می‌دهند و مکرراً این کار انجام می‌شود تا تمام مسیرهایی که وابسته به گره تازه از دست رفته بوده‌اند از کلّ جداول مسیریابی حذف شوند.
به عنوان مثالی از روش «نگهداری مسیر»، مثال قبلیمان را مدنظر قرار بدهید ولیکن فرض کنید که G به ناگاه خاموش شده است. توپولوژی تغییریافتهٔ شبکه در شکل ۵-۲۳-ب نشان داده شده است؛ وقتی D متوجه می‌شود که G از میان رفته است به جدول مسیریابی خود مراجعه می‌کند و می‌بیند که G بر روی مسیری قرار داشته که به E،

۱. قطر شبکه به معنای طول بزرگترین مسیر در شبکه است. -م

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(الف)



(ب)

شکل ۵-۲۳. (الف) جدول مسیریابی D قبل از آنکه G از کار بیفتد. (ب) گراف پس از حذف G از شبکه.

G و I ختم می‌شده است. اجتماع مجموعهٔ همسایه‌های فعال این سه گره مقصد عبارت است از {B و A}. به عبارت دیگر A و B در برخی از مسیرهای خود به G متکی بوده‌اند لذا باید به این مسیریابها اطلاع داد که G دیگر کار نمی‌کند. D با ارسال بسته‌های خاصی این موضوع را بدانها اطلاع داده تا آنها نیز جدول مسیریابی خود را اصلاح نمایند. خود D نیز درایه‌های متناظر با G ، E و I را از جدول مسیریابی خود حذف می‌کند.[1]

شاید از توضیحاتی که در خصوص الگوریتم AODV ارائه کردیم مشخص نشده باشد که بنیانی‌ترین تفاوت بین این الگوریتم و الگوریتم **بلمن-فورد** آن است که گره‌ها بطور متناوب کلّ جدول مسیریابی خود را منتشر نمی‌کنند، بلکه فقط بخش کوچکی از آنرا و آنهم بر حسب تقاضا، ارسال می‌نمایند. این ویژگی، در پهنای باند مصرفی و طول عمر باطری گره‌های متحرک صرفه جویی می‌کند.

AODV همچنین قادر به پخش داده‌های فراگیر (Broadcast) و مسیریابی چندپخشی (Multicast) است. برای آگاهی بیشتر به مرجع (Perkins and Royer,2001) مراجعه نمائید. مسیریابی در شبکه‌های ویژه، یک موضوع پژوهشی داغ و جدید است و اخیراً در این خصوص مقالاتی تألیف و منتشر شده است. برخی از آنها را می‌توان در مراجع ذیل یافت:

Chen et al., 2002; Hu and Johnson, 2001; Li et al., 2001; Raju and Garcia-Luna-Aceves, 2001; Ramanathan and Redi, 2002; Royer and Toh, 1999; Spohn and Garcia-Luna-Aceves, 2001; Tseng et al., 2001; and Zadeh et al., 2002).

۵.۲.۱۱ جستجوی گره در شبکه‌های همتا به همتا (Peer-to-Peer)

یکی از پدیده‌های نسبتاً جدید، شبکه‌های همتا به همتا است که در آن تعداد کثیری از افراد برای به اشتراک گذاشتن منابع مشترک خود، مستقیماً با یکدیگر در تماسند (این افراد عموماً از طریق یک اتصال ثابت و باسیم به اینترنت متصل شده‌اند). اولین کاربرد بسیار گسترده از تکنولوژی «همتا به همتا» به یکی از بحث‌برانگیزترین ماجراهای «گناه جمعی» بدل شد: ۵۰ میلیون از کاربران Napster می‌توانستند فایلهای موزیک و آواز را که حق امتیاز آنها در

---

۱. توضیحی برای رفع ابهام از مفهوم همسایه‌های فعال خالی از لطف نیست. به سطر آخر از جدول ۵-۲۳-ب دقت کنید. این جدول فرضاً متعلق به D است و آخرین رکورد آن بیان می‌کند که برای رسیدن به گره I باید بسته‌ها بـه G فـرستاده شـوند و تـا رسیدن به I فقط دو گام باقیمانده است؛ در فیلد «همسایه‌های فعال» نام A و B درج شده است یعنی: A و B برای ارسال بسته‌هایشان به سوی I آنها را به من -D- می‌دهند. لذا در صورت خرابی G باید به این دو همسایه اطلاع داد تا آنها نیز از این موضوع آگاه شده و جداول خود را اصلاح کنند. -م

تملک دیگران بود بدون اجازه از صاحبین آنها با یکدیگر رد و بدل کنند؛ نهایتاً Napster پس از مناقشات فراوان به حکم دادگاه تعطیل شد.[1] بهر تقدیر، تکنولوژی همتا به همتا کاربردهای قانونی بسیار جالبی دارد. این تکنولوژی با مسائل و مشکلات مسیریابی مواجه است؛ ولیکن این مسائل با آنچه که تاکنون مطالعه کرده‌ایم کاملاً یکسان نیست، لذا مروری اجمالی بر آن خالی از لطف نخواهد بود.

آنچه که سیستمهای «همتا به همتا» را جذّاب کرده آنست که سیستمی کاملاً توزیع‌شده ایجاد می‌کند یعنی تمام گره‌ها متقارن هستند و هیچگونه کنترل مرکزی یا سلسله‌مراتب خاصی بر آن حاکم نیست. در یک سیستم رایج همتا به همتا، هر یک از کاربران دارای مقداری اطلاعات هستند که ممکن است برای کاربران دیگر جالب باشد. این اطلاعات می‌تواند نرم‌افزارهای رایگان و عمومی، موزیک، عکس و نظایر آنها باشد. هر گاه تعداد کاربران زیاد باشد، یکدیگر را نمی‌شناسند و نمی‌دانند آنچه را که در جستجوی آن هستند در کجا پیدا کنند. یک راه حلّ آن است که یک پایگاه دادهٔ مرکزی و بسیار بزرگ از فهرست آنها داشته باشیم ولی ممکن است به دلایلی امکان‌پذیر نباشد (مثلاً هیچکس حاضر به میزبانی و نگهداری آن نشود) بنابراین مسئله اساسی آن است که در غیاب یک پایگاه مرکزی یا یک فهرست مرکزی، یک کاربر چگونه می‌تواند گره‌ای را که اطلاعات مورد نیاز او را در اختیار دارد، پیدا کند.

اجازه بدهید فرض را بر آن بگذاریم که هر کاربر یک یا چند آیتم داده مثل موزیک، عکس، برنامه، فایل و نظائر آنها در اختیار دارد و کاربران دیگر علاقمند به خواندن آنها هستند. هر آیتم توسط یک رشتهٔ ASCII نامگذاری شده است. کاربران احتمالی فقط همین رشتهٔ ASCII را می‌دانند و می‌خواهند بدانند آیا فردی یا افرادی آن را در اختیار دارند و اگر دارند آدرس IP آنان چیست؟

به عنوان مثال به پایگاه توزیع‌شدهٔ مردم‌شناسی (شجره‌نامهٔ مردم) دقت کنید. هر کسی که به مردم‌شناسی علاقمند است فرضاً در خصوص اجداد و بستگان خود اطلاعاتی مثل عکس، صدا یا حتی قطعات ویدیویی در اختیار دارد. ممکن است جدّ اعلای بسیاری از افراد یکی باشد و امکان دارد اطلاعات مربوط به آن در چندین «گره» وجود داشته باشد. نام هر رکورد عموماً نام فرد موردنظر (در شکل و قالب مشخص) است. حال در جایی یک نفر که علاقمند به تحقیق در مورد شجره‌نامهٔ خود بوده متوجه می‌شود که نام جدّ اعلای او در آرشیو یک گره خاص وجود دارد و ساعت جیبی و طلائی خود را برای برادرزاده‌اش به ارث گذاشته است!! حال او می‌خواهد بداند که نام این برادرزاده چیست و آیا در جایی دیگر رکوردی در این رابطه وجود دارد. بدون وجود یک پایگاه اطلاعات مرکزی، چگونه می‌توان کسی که چنین رکوردی را در اختیار دارد، پیدا کرد؟

برای حل این مسئله الگوریتمهای گوناگونی پیشنهاد شده است. الگوریتمی که بررسی می‌کنیم **«الگوریتم Chord»** نام دارد. (Dabek et al., 2001; Stoica et al., 2001) شرح سادهٔ عملکرد آن بدینگونه است: سیستم Chord از مجموعهٔ n کاربر شرکت‌کننده تشکیل شده است. هر یک از این کاربران ممکن است رکوردهایی را در خود ذخیره کرده و همچنین آمادگی داشته باشند بخشی از فهرست (ایندکس) سیستم را برای استفادهٔ دیگر کاربران نگهداری و عرضه کنند. هر یک از کاربران دارای یک آدرس IP هستند که این آدرس را می‌توان توسط یک **«تابع درهم‌سازی»** (Hash Function) به یک آدرس m بیتی تبدیل کرد. در سیستم Chord از تابع درهم‌سازی SHA-1 استفاده می‌شود؛ SHA-1 در مبحث رمزنگاری کاربرد دارد و در فصل هشتم نگاهی بدان خواهیم انداخت. فعلاً به همین اندازه بسنده می‌کنیم که این تابع یک رشته از بایتها با طول دلخواه را گرفته و براساس آن یک عدد ۱۶۰ بیتی

---

۱. در حقیقت کاری که Napster انجام می‌داد این بود که مشترکین خود را که فایلهای موزیک بر روی کامپیوتر خودداشتند بهم معرفی می‌کرد تا بتوانند به صورت بی‌واسطه و رو در رو (یعنی همان همتابه همتا) فایلهای خود را ردّ و بدل کنند. اینکار خشم دست‌اندرکاران صنعت موسیقی را برانگیخت! -م-

بشدّت تصادفی تولید می‌کند. بنابراین می‌توانیم یک آدرس IP را به عددی ۱۶۰ بیتی و یکتا تبدیل کنیم که به این عدد اصطلاحاً «**شناسهٔ گره**» (Node Identifier) گفته می‌شود.

از دیدگاه مفهومی تجسّم کنید که تمام 2^{160} شمارهٔ شناسهٔ گره‌ها به صورت صعودی پیرامون یک دایرهٔ بزرگ چیده شده‌اند. برخی از این شماره‌ها مربوط به گره‌های شرکت‌کننده (کاربران فعّال) هستند ولی بیشتر آنها پوچند. در شکل ۵-۲۴-الف ما دایرهٔ شماره‌ها را برای m=۵ نشان داده‌ایم. (فعلاً به کمانهای میانی کاری نداشته باشید). m=۵ بدین معناست که شماره‌ها پنج بیتی هستند و شماره‌های از ۰ تا ۳۱ را در بر می‌گیرند. در این مثال گره‌های ۱ ، ۴ ، ۷ ، ۱۲ ، ۱۵ ، ۲۰ و ۲۷ مربوط به گره‌های واقعی بوده و در شکل به صورت سایه‌دار نشان داده شده‌اند. مابقی شماره‌ها وجود خارجی ندارند. (شماره‌های پوچ)

تابع *successor(k)* را بدین مضمون تعریف می‌کنیم که اولین «شناسهٔ گره» مربوط به اولین کاربر واقعی که پس از عدد k قرار گرفته را بر می‌گرداند. به عنوان مثال $successor(6)=7$ ، $successor(8)=12$ و $successor(22)=27$.

اسامی رکوردها (مثل نام فایلهای آواز و نظائر آن) نیز توسط تابع درهم‌سازی (یعنی SHA-1) به یک رشته ۱۶۰ بیتی تبدیل می‌شوند که این رشته «**کلید**» (Key) نامیده می‌شود. لذا برای تبدیل یک نام (یعنی نام ASCII هر رکورد) به «**کلید**» متناظر با آن، رابطهٔ *Key=hash(name)* را تعریف کرده‌ایم. برای این کار فقط کافی است پروسیجر محلی *hash* را فراخوانی نماییم. اگر کسی که یک رکورد مفید با نام name در اختیار دارد، بخواهد آن را در دسترس عموم قرار بدهد ابتدا یک شاخص دوتایی شامل (IP Address,name) ساخته و برای ذخیره کردن این شاخص بر روی یک گره مناسب ، تابع *successor*(*hash*(name)) را فراخوانی می‌نماید. اگرچه ممکن است چندین رکورد با همین نام در گره‌های متفاوت موجود باشد ولیکن شاخص دوتایی آنها در یک گره ذخیره می‌شود.[1] بدین طریق، ایندکس رکوردها (شاخص دوتایی هر رکورد) به صورت تصادفی بر روی گره‌ها توزیع می‌شوند. برای آن که تحمل خرابی (Fault Tolerance) این سیستم بالا باشد می‌توان از تعداد p تابع مختلف hash استفاده کرد تا هر شاخص دوتایی در p گره مختلف ذخیره شود ولیکن فعلاً به این موضوع نخواهیم پرداخت.

اگر بعداً کاربری بخواهد آیتمی با نام *name* را جستجو نماید ابتدا آن را توسط تابع hash درهم‌سازی می‌کند تا کلید آن بدست آید. سپس تابع *successor*(key) را فراخوانی می‌کند تا آدرس IP گره‌ای که فهرست شاخصها را ذخیره کرده، پیدا شود. اگرچه اولین مرحله ساده است ولی مرحلهٔ دوم چندان ساده نیست. برای آن که بتوان آدرس IP گره‌ای را که متناظر با یک کلید خاص است پیدا کرد، هر گره باید چندین «**ساختمان دادهٔ نظارتی**»[2] در خود ذخیره نماید. یکی از این ساختمانهای داده، آدرس IP گره‌ای است که در دایرهٔ «شناسهٔ گره‌ها»، پس از خود گره قرار گرفته است. به عنوان مثال در شکل ۵-۲۴ ، گره بعدی ۴ ، گره ۷ است و گره بعدی ۷، گره ۱۲ است.

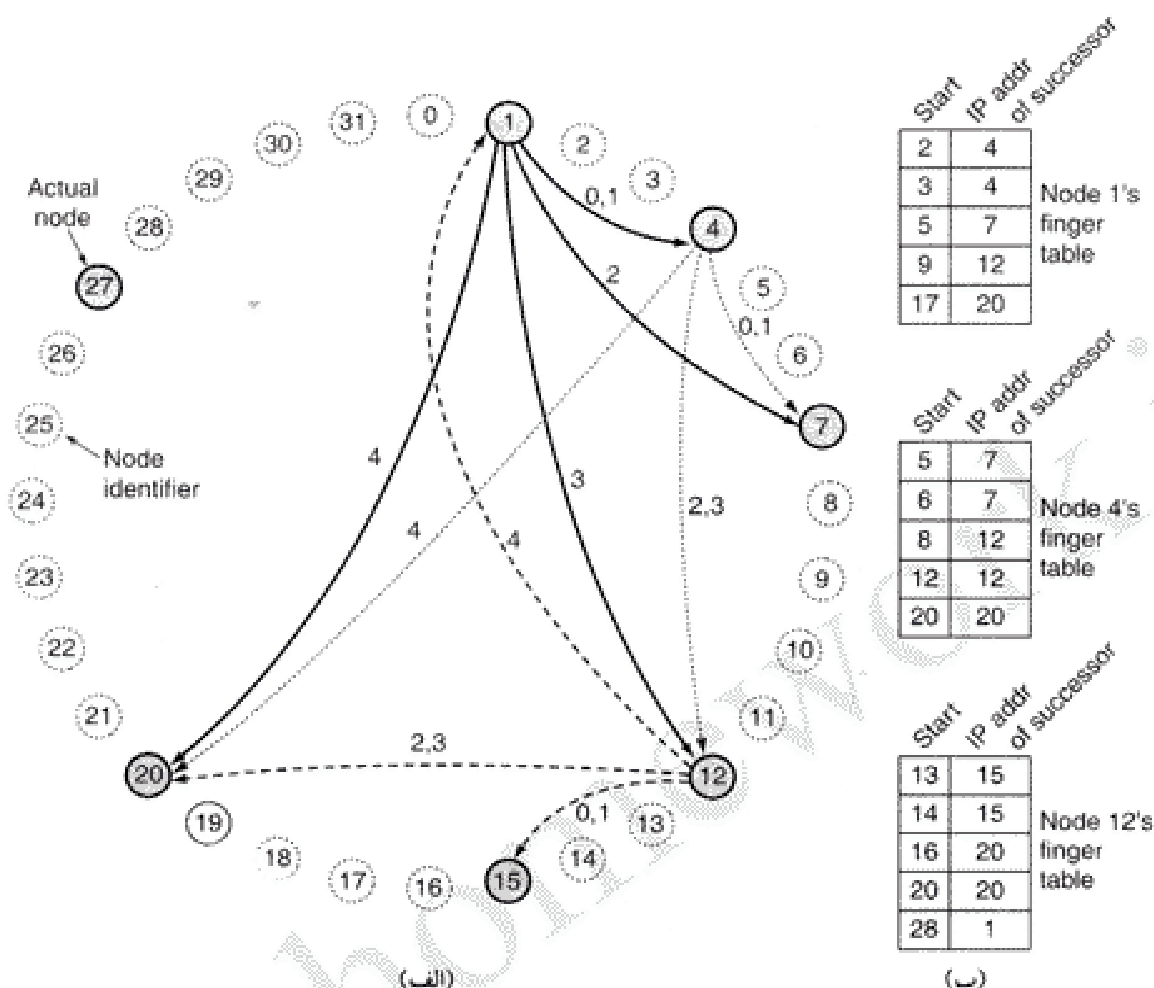
حال مراحل جستجو به ترتیب زیر ادامه می‌یابد: گره متقاضی بسته‌ای را برای گره بعدی خود ارسال کرده و ضمن اعلام آدرس IP خود، کلید مورد جستجو را مشخص می‌نماید. این بسته حول این دایره منتشر می‌شود تا آن که گره مورد جستجو پیدا شود. هر گره بررسی می‌کند که آیا اطلاعات مورد جستجو و منطبق با کلید را در اختیار دارد یا خیر. اگر داشته باشد آن اطلاعات را مستقیماً به آدرس IP متقاضی ارسال می‌نماید و در غیر اینصورت بستهٔ تقاضا را برای بعدی خود در دایره می‌فرستد.

اولین بهینه‌سازی این سیستم آن است که هر گره آدرس IP گره قبلی و بعدی خود را داشته باشد تا تقاضاها بتوانند در دو جهت ساعتگرد و پادساعتگرد (بسته به آن که کدام مسیر کوتاهتر است)، ارسال شوند. به عنوان مثال

---

۱. در حقیقت شناسنامهٔ هر رکورد مثل فایلهای صدا، تصویر یا برنامه ، که شامل نام آن و آدرس IP ماشین نگهدارندهٔ آنست، بر روی ماشینهای دیگری قرار می‌گیرد. -م

۲. Administrative Data Structure



Start	IP addr of successor
2	4
3	4
5	7
9	12
17	20

Start	IP addr of successor
5	7
6	7
8	12
12	12
20	20

Start	IP addr of successor
13	15
14	15
16	20
20	20
28	1

شکل ۵-۲۴. (الف) مجموعه‌ای از ۳۲ شناسهٔ گره که پیرامون یک دایره مرتب شده‌اند. گره‌های خاکستری رنگ متناظر با یک ماشین واقعی هستند. فلشها اشاره گرهای جدول Finger را نشان می‌دهند. برچسبهای هر فلش اندیسهای جدول Finger هستند. (ب) مثالی از جداول Finger

در شکل ۵-۲۴، گره ۷ می‌تواند برای یافتن گره ۱۰ در جهت ساعتگرد اقدام کند در حالی که برای یافتن گره ۳ می‌تواند به صورت پادساعتگرد عمل نماید.

حتی با وجود دو جهت برای جستجو، در سیستمهای عظیم «همتا به همتا» جستجوی خطی (Linear Search) بسیار ناکارآمد و کُند عمل می‌کند چرا که در هر جستجو بطور متوسط باید n/2 گره بررسی شود. برای بالا بردن سرعت جستجو، هر گره دارای جدولی است که در سیستم Chord اصطلاحاً Finger Table نامیده شده است. جدول Finger حاوی m درایه (Entry) است که از صفر تا m-1 شماره‌گذاری می‌شود و هر یک از آنها به آدرس گره‌های واقعی اشاره می‌کنند. هر یک از این درایه‌ها دارای دو فیلد هستند: فیلد start و آدرس IP گره‌ای که شناسهٔ آن معادل *successor*(start) است. در شکل ۵-۲۴-ب، جدول Finger برای سه گره نمونه نشان داده شده است. مقادیر فیلدهای مربوط به درایهٔ i در گره k عبارتست از:

$start = k + 2^i \quad (\text{modulo } 2^m)$

IP address of *successor*(*start*[i])

دقت کنید که هر گره، آدرس IP تعداد نسبتاً کمی از گره‌ها را در خود ذخیره می‌کند و اغلب این گره‌ها دارای شمارهٔ

شناسهٔ نزدیک بهم هستند. [یعنی در هر گره آدرس IP گره‌هایی نگهداری می‌شوند که بر روی دایرهٔ شناسه‌ها، تقریباً در کنار آن گره باشند. -م]

با استفاده از جدول Finger ، جستجوی کلید Key در گره k به ترتیب زیر انجام می‌شود: اگر مقدار key بین k و *successor*(k) باشد، گره‌ای که اطلاعاتی در خصوص key در اختیار دارد همان *successor*(k) است و جستجو خاتمه می‌یابد. در غیر اینصورت در جدول Finger جستجو می‌شود تا درایه‌ای که در آن، مقدار فیلد start ، به *predecessor*(k) [یعنی شناسهٔ گره قبلی k در دایره] نزدیکتر است پیدا شود. این تقاضا مستقیماً به آدرس IP گره فوق‌الذکر ارسال می‌شود تا ادامهٔ جستجو در آنجا پیگیری شود. (آدرس IP گره مذکور از جدول Figer بدست می‌آید.) از آنجایی شناسهٔ این گره نزدیک به key و در عین حال کمتر از آن است لذا این شانس خوب وجود دارد که با تعداد کمی پرس‌وجوی دیگر نتیجهٔ مورد نظر بدست آید. در حقیقت چون که در هر جستجو فاصلهٔ باقیمانده تا هدف مورد نظر نصف می‌شود می‌توان نشان داد که متوسط تعداد جستجوها log_2n است.

به عنوان اولین مثال فرض کنید در گره ۱ ، کلید key=3 را جستجو می‌کنیم. از آنجایی که گره ۱ می‌داند که ۳ بین خودش و گره بعدی او یعنی ۴ واقع شده، لذا گره مورد نظر همان ۴ است و جستجو خاتمه یافته و آدرس IP گره ۴ بدست می‌آید.

در مثال دوم، فرض کنید در جستجوی کلید key=14 بر روی گره ۱ هستیم. چون که ۱۴ بین ۱ و ۴ نیست، از جدول Finger کمک گرفته می‌شود. نزدیکترین عدد قبل از ۱۴ ، ۹ است بنابراین، تقاضا به سوی آدرس IP گره حقیقی پس از ۹ که در جدول ۱۲ تعیین شده است هدایت می‌شود. گره ۱۲ می‌بیند که گره ۱۴ بین خودش و گره بعدیش یعنی ۱۵ قرار گرفته ، لذا آدرس IP گره ۱۵ را برمی‌گرداند.

به عنوان مثال سوم، فرض کنید بر روی گره ۱ به دنبال کلید key=16 هستیم. مجدداً این تقاضا برای گره ۱۲ ارسال می‌شود ولی در اینجا گره ۱۲ پاسخ این تقاضا را نمی‌داند و به همین دلیل نزدیکترین شمارهٔ قبل از ۱۶ یعنی ۱۴ را یافته و از طریق جدول خود آدرس IP گره ۱۵ را بدست می‌آورد؛ سپس تقاضا برای آن گره ارسال می‌شود. گره ۱۵ می‌بیند که عدد ۱۶ بین خودش و گره بعدی یعنی ۲۰ قرار دارد لذا آدرس IP گره ۲۰ را به سوال‌کننده بر می‌گرداند و به همین ترتیب کار ادامه می‌یابد تا به گره ۱ برسد.

از آنجایی که گره‌ها بطور متوالی به شبکه می‌پیوندند یا از آن جدا می‌شوند فلذا Chord نیازمند روشی است که بتواند این مسئله را نیز حل و فصل کند. فرض را بر آن می‌گذاریم که وقتی این سیستم آغاز به کار کرده تعداد گره‌ها آنقدر کم بوده که توانسته‌اند مستقیماً با یکدیگر مبادلهٔ اطلاعات کرده و اولین دایره و جداول Finger را بسازند. پس از آن ، به یک روال خودکار نیاز است: وقتی گره جدید r می‌خواهد به این سیستم بپیوندد باید با یکی از گره‌های موجود تماس برقرار کرده و از او بخواهد که آدرس IP گره *successor*(r) را برایش پیدا کند. پس از پیدا شدن، گره جدید از *successor*(r) می‌خواهد که گره قبلی خود خود را معرفی کند. در آخر، گره جدید از این دو گره می‌خواهد که او را به عنوان گره r در دایره وارد کند. به عنوان مثال اگر در شکل ۵-۲۴ گره ۲۴ بخواهد به سیستم بپیوندد از یکی از گره‌های فعال تقاضا می‌کند که *successor*(24) را برایش پیدا نماید که در این جا ۲۷ است. سپس از ۲۷ در مورد گره ماقبل او یعنی ۲۰ سؤال می‌کند. پس از آن که حضور خود را به این دو گره (یعنی ۲۰ و ۲۷) اعلام کرد گره ۲۰ گره ۲۴ را به عنوان گره بعدی خود و گره ۲۷ آن را به عنوان گره قبلی خود به رسمیت می‌شناسند. مضاف بر این، گره ۲۷ آن دسته از درایه‌هایی را که کلیدشان در محدودهٔ ۲۱ تا ۲۴ است به گره ۲۴ تقدیم می‌دارد. در این لحظه گره ۲۴ بطور کامل به جمع پیوسته است. ولیکن در این لحظه بسیاری از جداول Finger غلط هستند. برای اصلاح این جداول، هر گره یک پروسه را که در پس‌زمینه اجرا می‌شود ، اجرا می‌کند تا جداول مربوطه (با فراخوانی متناوب تابع *successor*) از نو محاسبه و اصلاح شوند. هر گاه در خلال عملیات تازه‌سازی، یکی از این

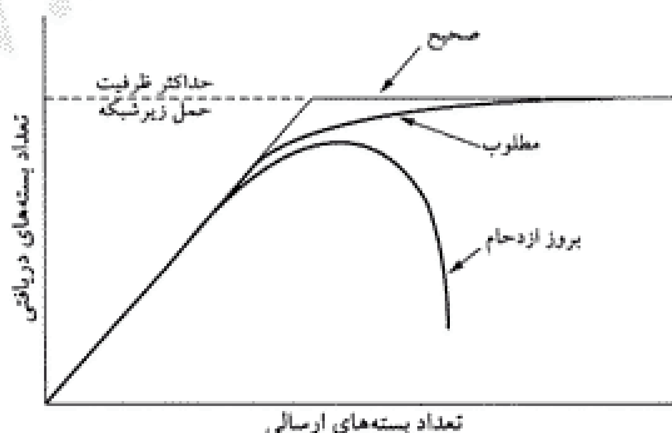
تقاضاها به گره جدیدی بر بخورد درایهٔ مربوط به آن نیز در جدول Finger بهنگام می‌شود.

وقتی یک گره به صورت مسالمت‌آمیز سیستم را ترک می‌کند کلیدهای خود را به گره بعدی خود در حلقه تسلیم می‌کند تا به او اطلاع بدهد که در آستانهٔ خروج از سیستم است و آن گره بتواند پیوند خود را با گره ماقبل از گره در آستانهٔ خروج، برقرار نماید. وقتی گره‌ای به ناگاه از کار بیفتد مشکل پیش می‌آید چراکه گره ماقبل از گره خراب شده، هیچ گره معتبر و فعال بعدی ندارد. برای کاهش اثر این مشکل، هر گره نه تنها آدرس مستقیم گره بعدی خود را نگه می‌دارد بلکه آدرس مستقیم s گره بعد از خود را نیز دارد تا حتی اگر s-1 گره متوالی از کار بیفتد باز هم بتوان دایره را اصلاح کرد.

از سیستم Chord برای ایجاد سیستم توزیع شدهٔ فایل (Dabek et al., 2001) و چند کاربرد دیگر استفاده شده و تحقیقات بر روی آن کماکان ادامه دارد. یک سیستم همتابه همتای دیگر به نام Pastry و کاربردهای آن در مرجع (Rowstron and Druschel, 2001) تشریح شده است. سیستم سومی به نام Freenet نیز معرفی شده که مستندات آن در (Clark et al., 2002) در دسترس عموم قرار دارد. چهارمین سیستم از این نوع نیز در مرجع (Ratnasamy,2001) تشریح شده است.

۵-۳ الگوریتمهای کنترل ازدحام

وقتی به بخشی از زیرشبکه، تعداد بسیار زیادی بسته تحویل شود کارآیی آن کاهش می‌یابد. بدین وضعیت **«ازدحام»** (Congestion) گفته می‌شود. شکل ۵-۲۵ ، علائم بروز این وضعیت را نشان می‌دهد. هر گاه تعداد بسته‌هایی که توسط ماشینهای میزبان به زیرشبکه سرازیر می‌شوند متناسب با ظرفیت حمل زیرشبکه باشد تمام این بسته‌ها تحویل مقصدشان خواهند شد (به استثنای آنهایی که در اثر خطای انتقال آسیب می‌بینند) و تعداد بسته‌های تحویلی متناسب با تعداد بسته‌های ارسالی است. ولیکن به محض افزایش بی‌رویهٔ ترافیک، مسیریابها قادر نیستند از عهدهٔ آن بر آمده و بسته‌ها شروع به از دست رفتن می‌کنند. این مسئله حادّتر نیز می‌شود و در ترافیک بسیار بالا کارآیی بطور کامل سقوط کرده و هیچ بسته‌ای تحویل مقصد نخواهد شد!



شکل ۵-۲۵. وقتی ترافیک تحویلی به شبکه بیش از اندازه باشد ازدحام بوجود می‌آید و کارآیی بشدت افت می‌کند.

چندین عامل می‌تواند به بروز ازدحام بینجامد. اگر به ناگاه دنباله‌ای از بسته‌ها بر روی سه یا چهار خط ورودی دریافت شده و خط خروجی همهٔ آنها یکی باشد صف تشکیل خواهد شد و اگر فضای حافظهٔ کافی برای نگهداری تمام آنها وجود نداشته باشد، بسته‌ها از بین می‌روند. شاید اضافه کردن حافظه مفید به نظر برسد ولیکن «ناگل» (Nagle,1987) بدین نتیجه رسید که حتی اگر حافظهٔ مسیریاب نامحدود باشد، وضعیت ازدحام نه تنها بهتر

نمی‌شود بلکه بدتر هم خواهد شد زیرا بسته‌ها، زمانی به سر صفّ می‌رسند که مهلت رسیدنشان به مقصد تمام شده و نسخه‌های تکراری آن ارسال شده‌اند. تمام این بسته‌های تکراری نیز برحسب وظیفه‌شناسی به مسیریاب بعدی هدایت شده و در تمام مسیر رسیدن به مقصد، «بار» افزایش می‌یابد. [بعبارت دیگر، «ازدحام» قابلیت انتشار در کل یک مسیر دارد.]

پردازنده‌های کُند نیز می‌توانند عامل بروز ازدحام باشند. اگر پردازندهٔ اصلی مسیریاب در انجام وظائف محوله به خود (مثل عملیات صف‌بندی، بهنگام سازی جداول مسیریابی و نظائر آن) کُند عمل نماید، صف ایجاد می‌شود، حتی وقتی که ظرفیت خطوط خروجی بیش از حدّ مورد نیاز است.

به دلیل مشابه، خطوط با پهنای باند کم نیز می‌توانند منجر به بروز ازدحام شوند. ارتقاء پهنای باند خطوط بدون تغییر در پردازنده‌ها یا بالعکس اغلب فایدهٔ چندانی ندارد و فقط جای گلوگاه و منشاء مشکل را تغییر می‌دهد. همچنین ارتقاء کامل یک بخش کوچک [مثل تغییر یک مسیریاب و ارتقاء ظرفیت خطوط آن] بدون تغییر در کلّ سیستم، فقط محل بروز مشکل و گلوگاه را جابجا می‌کند و در بهبود کارآیی کل سیستم تأثیر چشمگیری نخواهد داشت. مشکل اساسی یک سیستم، عدم تطابق و تناسب بخشهای مختلف آنست. این مشکل تا زمانی که کلیهٔ مؤلفه‌های سیستم متعادل نشوند باقی خواهد ماند.

باید به صراحت تفاوت بین «**کنترل ازدحام**»[1] و «**کنترل جریان**»[2] و همچنین ارتباط ظریف این دو مکانیزم را تبیین کنیم. «کنترل ازدحام» مکانیزمهایی است «جهت ایجاد اطمینان از این که زیرشبکه قادر به حمل ترافیک عرضه شده به آن هست». این مشکل یک مورد همگانی و سراسری است و از رفتار و عملکرد تمام ماشینهای میزبان، کلّیه مسیریابها، عملیات پردازشی «ذخیره و هدایت» (Store & Forward) درون مسیریابها یا هر عامل دیگری که ظرفیت حمل زیرشبکه را کاهش بدهد، ناشی می‌شود.

در مقابل، «**کنترل جریان**» به ترافیک نقطه‌به‌نقطه بین یک فرستنده و گیرندهٔ مفروض مربوط می‌شود و وظیفهٔ اصلی آن ایجاد اطمینان از این موضوع است که یک فرستندهٔ سریع نمی‌تواند متوالیاً داده‌ها را با سرعتی بیش از توانایی دریافت گیرنده، ارسال نماید. در کنترل جریان، گزارشات و فیدبکهای مستقیمی از گیرنده به فرستنده ارسال می‌شود تا به فرستندهٔ طرف مقابل تفهیم کند که کارها را چگونه انجام بدهد.

برای درک اختلاف بین این دو مفهوم، یک شبکهٔ فیبرنوری با ظرفیت 1000 Gagabits/sec را در نظر بگیرید که در آن یک اَبَرکامپیوتر سعی می‌کند با سرعت یک گیگابیت بر ثانیه فایلی را برای یک کامپیوتر شخصی، ارسال کند! اگرچه هیچگونه ازدحامی رخ نخواهد داد (خود شبکه مشکلی ندارد) ولیکن برای آن که بتوان به نحوی اَبَرکامپیوتر را وادار کرد که هرازگاهی متوقف شود و اجازهٔ نفس کشیدن به کامپیوتر شخصی بدهد به مکانیزمهای کنترل جریان نیاز است.

در سمت مقابل، یک شبکهٔ «ذخیره و هدایت» با خطوط 1Mbps و هزار کامپیوتر بزرگ را در نظر بگیرید که در آن نیمی از کامپیوترها سعی می‌کنند با سرعت 100Kbps برای نیم دیگر، فایل ارسال کنند! در اینجا مشکل تحت فشار قرار گرفتن گیرندهٔ کُند توسط فرستندهٔ سریع مطرح نیست بلکه مسئله آنست که ترافیک تحمیل شده به شبکه از میزان ظرفیت و توانایی آن بیشتر است.

دلیل آن که مفهوم کنترل ازدحام، اغلب با مفهوم کنترل جریان اشتباه می‌شود آن است که در برخی از الگوریتمهای کنترل ازدحام، هنگام بروز مشکل برای شبکه، پیغامهایی را برای ماشینهای مبداء ارسال کرده و به آنها تفهیم می‌کند که باید سرعت خود را پایین بیاورند. بنابراین یک ماشین میزبان ممکن است به دو دلیل پیغام

---
۱. Congestion Control	۲. Flow Control

Slowdown (آهسته‌تر ارسال کن) را دریافت کند: اول آن که گیرنده نتواند با همان سرعتی که فرستنده ارسال می‌کند داده‌ها را دریافت نماید. دوم آن که شبکه با ازدحام مواجه شود و از عهدهٔ ارسال داده‌ها بر نیاید. بعداً باز هم به این موضوع بر می‌گردیم.

مطالعات خود پیرامون کنترل ازدحام را با نگاهی به مدل و روشهای عمومی برخورد با آن آغاز می‌کنیم. سپس به راهکارهای گسترده‌ای خواهیم پرداخت که سعی می‌کنند در همان ابتدا از بروز این مشکل پیشگیری کنند. همچنین الگوریتمهای پویایی را مرور می‌کنیم که پس از بروز مشکل ازدحام، سعی در حلّ و فصل آن می‌کنند.

۵-۳-۱ اصول کلّی در کنترل جریان

بسیاری از مشکلات و مسائل سیستمهای پیچیده مثل شبکه‌های کامپیوتری را می‌توان از دیدگاه نظریهٔ کنترل بررسی کرد. در این روش تمام راه حلها به دو گروه تقسیم می‌شوند: «حلقه باز» و «حلقه بسته»[۱]. راه‌حلهای «حلقه باز» سعی می‌کنند یک مسئله را با طراحی خوب حل کرده و از همان ابتدا اطمینان بدهند که مشکلی رخ نخواهد داد. وقتی این سیستم شروع به کار و انجام فعالیت نمود هیچگونه نظارت یا تصحیح عملکرد ممکن نیست.

تمهیداتی که برای کنترل ازدحام به شیک «حلقه باز» پیش‌بینی شده عبارتند از: تصمیم‌گیری در خصوص زمان پذیرش ترافیک جدید، تصمیم‌گیری در خصوص زمان حذف بسته‌ها، انتخاب بسته‌هایی که باید حذف شوند و تصمیم‌گیری در خصوص زمان‌بندی صحیح در شبکه؛ حقیقت مشترک در تمام این موارد آنست که تصمیم‌گیری آنها مبتنی بر شرایط جاری شبکه نیست.

در مقابل، راه حلهای «حلقه بسته» مبتنی بر مفهوم حلقه‌های فیدبک هستند. اینگونه راهکارهای کنترل ازدحام، سه بخش را در بر می‌گیرند:

۱. نظارت بر سیستم به منظور تشخیص آنکه در کجا و چه وقت ازدحام رخ داده است.
۲. تحویل این اطلاعات به محلی که بایستی واکنش نشان بدهد.
۳. تنظیم عملکرد سیستم برای رفع مشکل

برای نظارت و تشخیص بروز ازدحام در زیرشبکه می‌توان معیارهای گوناگونی را بکار گرفت. مهمترین آنها عبارتند: درصد بسته‌هایی که به دلیل فقدان فضای کافی بافر حذف می‌شوند، متوسط طول صف، تعداد بسته‌هایی که مهلت ارسال آنها منقضی شده و از نو ارسال گردیده‌اند، متوسط تأخیر بسته و انحراف معیار[۲] تأخیر بسته. در تمام این معیارها رشد اعداد نمایانگر افزایش ازدحام است.

دومین مرحله از «حلقه فیدبک» آن است که اطلاعاتی در خصوص ازدحام، از محل تشخیص و بروز آن به محلی که می‌تواند کاری برای حلّ آن انجام بدهد، انتقال یابد. بدیهی‌ترین روش آن است که مسیریاب کشف‌کنندهٔ ازدحام بسته‌ای خاصّ برای ماشین یا ماشینهای مبداء بفرستد و مشکل را به آنها اعلام نماید. البته این بسته‌های اضافی خودشان به بار شبکه می‌افزایند، آن هم دقیقاً زمانی که شبکه ظرفیت هیچ بار اضافی ندارد یعنی دقیقاً حین ازدحام در زیرشبکه!

با این وجود روشهای دیگری نیز امکان‌پذیر است. به عنوان مثال می‌توان یک بیت یا یک فیلد در هر بسته کنار گذاشت تا هر گاه ازدحام از یک حد آستانه فراتر رفت، مسیریاب آنرا پر کند. هر گاه مسیریاب تشخیص بدهد که ازدحام رخ داده در تمام بسته‌های خروجی، این فیلد را پر می‌کند تا به همسایه‌های خود در این خصوص هشدار بدهد.

راهکار دیگر آن است که ماشینهای میزبان یا مسیریابها بطور متناوب بسته‌های «**آزمون**» تولید و ارسال کرده و

---

۱. Open loop , Close loop
۲. Standard deviation

مستقیماً در خصوص ازدحام کسب آگاهی کنند. بکمک این اطلاعات می‌توان بسته‌ها را بنحوی مسیریابی کرد که از ناحیهٔ بروز مشکل، ردّ نشوند. به مثابهٔ ایستگاههای کنترل ترافیک که هلی‌کوپترهای آنها بر روی شهرها به پرواز درمی‌آیند تا به شنودگان در حال حرکت هشدار بدهند تا بسوی نقاط بحران، حرکت نکنند.

در تمام روشهای مبتنی بر فیدبک، انتظار می‌رود که آگاهی از ازدحام موجب شود ماشینهای میزبان برای کاهش ازدحام از خود واکنش مناسبی نشان بدهند. برای آنکه این روشها به درستی کار کنند بایستی مقیاس زمان به دقت تنظیم شده باشد. اگر وقتی دو بستهٔ پشت سر هم می‌رسند مسیریاب فریاد بزند «ایست» و به محض آنکه فقط برای ۲۰ میکروثانیه آزاد می‌شود فریاد بزند: «حرکت»، سیستم بشدّت نوسانی عمل می‌کند و هیچگاه همگرا و پایدار نخواهد شد. برعکس اگر قبل از اعلام هر چیزی برای اطمینان به مدت ۳۰ دقیقه صبر کند، مکانیزم کنترل ازدحام آنقدر کُند واکنش نشان می‌دهد که عملاً هیچ سودی در دنیای واقعی نخواهد داشت. برای آنکه عملکرد خوبی انتظار داشته باشیم به حالت میانه‌ای نیازمندیم ولیکن بدست آوردن «**ثابت زمانی سیستم**» (Time Constant) مسئلهٔ چندان ساده‌ای نیست.

الگوریتمهای متعددی برای کنترل ازدحام معرفی شده‌اند. برای آنکه بتوان این الگوریتمها را به روش قابل فهمی سازماندهی کرد، Yang و Reddy (۱۹۹۵) الگوریتمهای کنترل ازدحام را طبقه‌بندی کرده‌اند. آنها به همان ترتیبی که در بالا تشریح شد تمام الگوریتمها را به دو ردهٔ «حلقه باز» و «حلقه بسته» تقسیم‌بندی نمودند. مضاف بر این، الگوریتمهای «حلقه باز» را برحسب اینکه در ماشینهای مبداء عمل می‌کنند یا در ماشینهای مقصد به دو رده تقسیم کردند. الگوریتمهای «حلقه بسته» نیز به دو ردهٔ فرعی تقسیم شده‌اند: «الگوریتمهای مبتنی بر فیدبک مستقیم» و «الگوریتمهای مبتنی بر فیدبک ضمنی». در الگوریتمهای مبتنی بر فیدبک مستقیم، برای هشدار دادن به مبداء، بسته‌های فیدبک دقیقاً از محل بروز ازدحام برگردانده می‌شوند. در الگوریتمهای مبتنی بر فیدبک ضمنی، مبداء با استناد به برخی از مشاهدات و علائم محلّی (همانند زمان لازم برای برگشت بسته‌های ACK) بروز ازدحام را حدس می‌زند.

بروز ازدحام بدین معناست که «بار شبکه» موقتاً از «منابع» موجود در برخی از بخشهای سیستم بیشتر شده و شبکه از عهدهٔ این بار بر نمی‌آید. [منظور از منابع پهنای باند، توان پردازشی CPU، حافظهٔ مسیریاب و نظایر آنهاست. -م] دو راه حلّ به ذهن متبادر می‌شود: افزایش منابع یا کاهش بار. به عنوان مثال زیرشبکه می‌تواند برای افزایش پهنای باند بین دو نقطه، موقتاً از یک خطّ تلفن (Dialup) کمکی بهره بگیرد. در سیستمهای ماهواره‌ای، افزایش توان فرستنده اغلب پهنای باند را افزایش می‌دهد. در ضمن اگر بجای استفادهٔ دائمی از بهترین مسیر، بخشی از بار بر روی مسیرهای دیگر تقسیم شود پهنای باند شبکه بطور مؤثری افزایش می‌یابد. نهایتاً آنکه می‌توان مسیریابهای یدکی را که فقط به عنوان پشتیبان در شبکه نصب شده‌اند (برای آنکه سیستم تحمل خرابی داشته باشد) به خدمت گرفت تا در هنگام بروز ازدحام، ظرفیت زیرشبکه افزایش یابد.

علیرغم این راهکارها، بسیاری از اوقات افزایش ظرفیت زیرشبکه ممکن نیست یا آنکه این ظرفیت تا آخرین حدّ افزایش یافته است. تنها راه باقیمانده برای رفع ازدحام کاهش بار است. چندین روش برای کاهش بار وجود دارد که از آن جمله می‌توان به این موارد اشاره کرد: (۱) اجتناب از سرویس دادن به برخی از کاربران (۲) کاهش سطح سرویس‌دهی به برخی یا تمام کاربران (۳) مجبور کردن کاربران به زمان‌بندی تقاضاهای خود به روشی قابل پیش‌بینی.[1]

---

۱. از آنجایی که برخی از ماشینها ترافیک انفجاری و غیرقابل پیش‌بینی تولید می‌کنند لذا می‌توان آنها را وادار کرد تا بکمک مکانیزمهای خاصّ (همانند مکانیزم بافرینگ یا مکانیزمهای شکل‌دهی به ترافیک) حجم ترافیک خود را متعادل و قابل پیش‌بینی کنند. -م

برخی از این روشها که به اختصار آنها بررسی خواهیم کرد در زیرشبکه‌های نوع «مدار مجازی» به بهترین نحو قابل اعمال هستند. در زیرشبکه‌های مدار مجازی، این روشها در لایهٔ شبکه پیاده می‌شوند. در زیرشبکه‌های دیتاگرام علیرغم اصراری که بر تفکیک وظائف لایه‌ها وجود دارد بخشی از این روشها باید در اتصالات لایهٔ انتقال بکار گرفته شوند. در این فصل به کاربرد این روشها در لایهٔ شبکه می‌پردازیم و در فصل بعدی به عملیات مدیریت ازدحام در لایهٔ انتقال، نگاهی خواهیم انداخت.

۵-۳-۲ سیاستهای پیشگیری از ازدحام

اجازه بدهید مطالعهٔ روشهای کنترل ازدحام را با بررسی سیستمهای «حلقه باز» شروع کنیم. اینگونه سیستمها به گونه‌ای طراحی می‌شوند تا بجای آنکه بگذارند ازدحام اتفاق بیفتد و بعداً واکنش نشان بدهند، در همان ابتدا ازدحام را به حداقل برسانند. این روشها برای رسیدن بدین هدف، از سیاستهای خاصّ و مناسب در سطوح مختلف، بهره می‌گیرند. در شکل ۵-۲۶ سیاستهای مختلفی که در لایه پیوند داده، لایهٔ شبکه و لایهٔ انتقال، می‌تواند بر پدیدهٔ ازدحام تأثیر بگذارد، فهرست شده‌اند. (Jain,1990)

لایه	سیاستها
انتقال	• سیاستهای ارسال مجدد • سیاستهای ذخیره بسته‌هایی که خارج از ترتیب می‌رسند. • سیاستهای تصدیق وصول بسته‌ها (Ack) • سیاستهای کنترل جریان • تعیین زمان انقضای مهلت تایمرها
شبکه	• مدار مجازی در مقابل روش دیتاگرام در زیرشبکه • مکانیزمهای صف‌بندی بسته‌ها و روشهای متنوع سرویس‌دهی • سیاستهای حذف بسته • الگوریتم مسیریابی • مدیریت طول عمر بسته‌ها
پیوند داده	• سیاستهای ارسال مجدد • سیاستهای ذخیره بسته‌هایی که خارج از ترتیب می‌رسند. • سیاستهای تصدیق وصول بسته‌ها (Ack) • سیاستهای کنترل جریان

شکل ۵-۲۶. سیاستهایی که بر پدیدهٔ ازدحام تاثیر می‌گذارند.

اجازه بدهید از لایهٔ پیوند داده شروع کرده و سپس به سمت بالا حرکت کنیم. سیاستهای ارسال مجدد (Retransmission Policy) با مسائلی از این قبیل سر و کار دارد: (۱) با چه سرعتی مهلت فرستنده (برای دریافت Ack یک بسته) خاتمه می‌یابد؟ (۲) در اثر انقضای مهلت چه چیزی ارسال می‌شود؟

یک فرستندهٔ عجول که مهلت آن به سرعت خاتمه می‌یابد و مبتنی بر پروتکل Go Back n تمام بسته‌های ارسالی را از نو می‌فرستد، بار بسیار سنگینتری را نسبت به فرستنده‌ای که از روش Selective Repeat استفاده کرده به زیرشبکه تحمیل می‌کند. یکی دیگر از مسائل که ارتباط تنگاتنگی با این موضوع دارد **سیاستهای بافرینگ** است. اگر گیرنده، بسته‌هایی را که خارج از ترتیب می‌رسند دور بینداز د تمام آنها باید از نو ارسال شوند و بار اضافی به زیرشبکه تحمیل خواهد شد. از دیدگاه کنترل ازدحام، روش «تکرار انتخابی» (Selective Repeat) به وضوح بهتر از روش Go Back n عمل می‌کند.

سیاست تصدیق دریافت بسته‌ها (Acknowledgement Policy) نیز بر روی ازدحام تأثیر می‌گذارد. اگر هر

بسته فوراً اعلام وصول شود، بسته‌های اعلام وصول (Ack) ترافیک زائد و اضافی تحمیل خواهند کرد. در عوض اگر برای صرفه‌جویی، از روش Piggybacking استفاده شود و اعلام وصول داده‌ها از طریق ترافیک معکوس انجام شود ممکن است مهلت فرستندهٔ داده‌ها متوالیاً منقضی شود و تکرار ارسال رخ بدهد که آن هم منجر به تحمیل بار اضافه خواهد شد. استفاده از یک روش محکم و سختگیرانه در الگوی کنترل جریان (مثلاً داشتن پنجرهٔ کوچک) می‌تواند نرخ ارسال داده را کاهش داده و به کاهش ازدحام کمک کند.

در لایهٔ شبکه، انتخاب بین روش «مدار مجازی» یا «دیتاگرام» تأثیر مستقیمی بر پدیدهٔ ازدحام دارد چراکه بسیاری از الگوریتمهای کنترل ازدحام فقط در زیرشبکه‌های مدار مجازی قابل‌اعمال هستند. صف‌بندی بسته‌ها و سیاستهای سرویس‌دهی نیز در کنترل ازدحام موثرند. این سیاستها بدین موضوع مرتبطند که آیا مسیریاب به ازای هر خط ورودی یک صف دارد، به ازای هر خط خروجی یک صف دارد یا هر دو صف را تشکیل می‌دهد. همچنین ترتیب پردازش بسته‌ها (مثلاً نوبت‌بندی چرخشی ـRound Robinـ یا روش مبتنی بر اولویت‌بندی) جزو سیاستها صف‌بندی محسوب می‌شود. «سیاست حذف بسته‌ها» قاعده‌ای است که براساس آن تعیین می‌شود که وقتی فضایی برای نگهداری بسته‌ها نیست کدامیک از بسته‌ها را باید حذف کرد. اتخاذ یک سیاست مناسب می‌تواند به کاهش ازدحام کمک کند و سیاستهای غلط شرایط ازدحام را بدتر خواهد کرد.

یک الگوریتم مسیریابی خوب می‌تواند با توزیع مناسب ترافیک بین خطوط مختلف به پیشگیری از ازدحام کمک کند در حالی که یک الگوریتم بد می‌تواند با ارسال ترافیک بسیار زیاد بر روی یک خط که با ازدحام مواجه شده شرایط را بدتر کند. مدیریت طول عمر بسته (Packet Lifetime) با این مسئله سر و کار دارد که چه مدت طول می‌کشد تا بسته [در صورت نرسیدن به مقصد] حذف شود. اگر این زمان بیش از حد طولانی باشد بسته‌های گمشده و سرگردان ممکن است تا قبل از موعد حذف شدن، شبکه را با انباشتگی و ازدحام مواجه کنند، در حالی که اگر این زمان کوتاه در نظر گرفته شود ممکن است بسته‌ها قبل از آنکه فرصت رسیدن به مقصد را داشته باشند حذف شوند و به تکرار ارسال بینجامد.

در لایهٔ انتقال همان موارد و سیاستهایی که در لایهٔ پیوند داده وجود داشت دنبال می‌شود ولی مضاف بر آنها روش تعیین بازه‌های زمانی انقضای مهلت (Timeout Interval) پیچیده‌تر است چراکه زمان عبور بسته‌ها از میان یک شبکه [با تعدادی مسیریاب و کانالهای متعدد] در مقایسه با عبور بسته‌ها از یک سیم واحد، چندان قابل پیش‌بینی نیست. اگر بازه‌های انقضای مهلت بیش از اندازه کوتاه باشد بسته‌های بی‌مصرف زیادی ارسال خواهد شد. اگر این زمان بیش از اندازه طولانی باشد، ازدحام کاهش می‌یابد ولیکن وقتی بسته‌ای به هر دلیل از دست برود زمان پاسخ [یعنی تأخیر ارسال مجدّد] مشکل‌آفرین خواهد شد.

۳-۳-۵ کنترل ازدحام در زیرشبکه‌های مدار مجازی

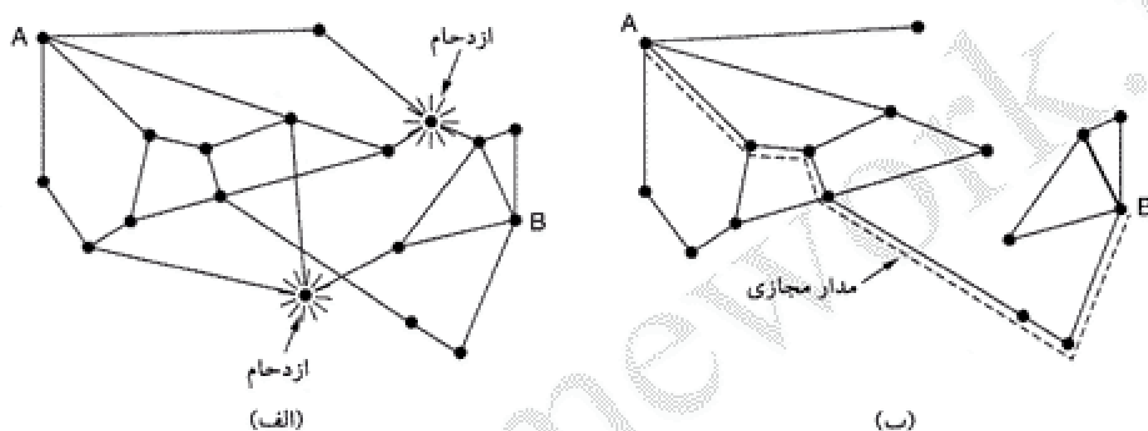
روشهای کنترل ازدحام که در بالا معرفی شدند اساساً «حلقه باز» هستند یعنی تلاش می‌کنند در همان ابتدا و قبل از بروز، از ازدحام پیشگیری کنند. در این بخش راهکارهایی را بررسی خواهیم کرد که در زیرشبکه‌های مدار مجازی به صورت پویا ازدحام را کنترل می‌کنند. در دو بخش آتی نیز نگاهی به تکنیکهای قابل استفاده در هر نوع زیرشبکه خواهیم انداخت.

یکی از تکنیکهایی که بطور گسترده‌ای از آن در جهت پیشگیری از وخیم شدن مشکل ازدحام (که از قبل شروع شده) استفاده می‌شود روش **«کنترل پذیرش»**[1] (Admission Control) نام دارد. ایدهٔ این روش ساده است: هر گاه ازدحام گزارش شود تا رفع کامل مشکل، هیچگونه تقاضای تنظیم مدار مجازی پذیرفته نخواهد شد. بدین

۱. بمعنای محدود کردن پذیرش ارتباط و امتناع از ایجاد مدار مجازی جدید

ترتیب هر گونه تلاش برای ایجاد اتصال در لایهٔ انتقال با شکست مواجه خواهد شد چراکه پذیرش افراد جدید وضع ازدحام را وخیم‌تر خواهد کرد. هر چند این روش خام و ناشیانه است ولی در عوض ساده و سهل‌الاجراست. در سیستمهای تلفن نیز هر گاه یک سوئیچ با بار بسیار زیاد مواجه شود، با قطع بوق آزاد (Dial Tone)، پذیرشهای جدید را محدود می‌نماید.

راهکار دیگر آن است که اجازهٔ تنظیم مدار مجازی جدید داده شود ولیکن مسیرهای انتخابی برای مدارات مجازی جدید در خارج از نواحی بحران‌زده انتخاب گردد. به عنوان مثال زیرشبکه شکل ۵-۲۷-الف را در نظر بگیرید که در آن دو مسیریاب با ازدحام مواجه شده‌اند.



شکل ۵-۲۷. (الف) یک زیرشبکهٔ مواجه با ازدحام (ب) ترسیم مجدد گراف زیرشبکه پس از حذف مناطق درگیر ازدحام. مدار مجازی مناسب بین A و B نشان داده شده است.

فرض نمایید که ماشین میزبان متصل به مسیریاب A بخواهد یک «اتصال» با ماشین متصل به B برقرار نماید. طبیعتاً این اتصال از یکی از مسیریابهای دارای مشکل عبور خواهد کرد. برای اجتناب از این وضعیت می‌توانیم زیرشبکه را مطابق با شکل ۵-۲۷-ب از نو ترسیم کرده و مسیریابهای دچار ازدحام و تمام خطوط آنها را حذف نماییم. خط نقطه‌چین، مسیر ممکن برای ایجاد مدار مجازی بین A و B را نشان می‌دهد. در این مسیر، مسیریابها و خطوط دچار ازدحام وجود ندارد.

استراتژی دیگر در زیرشبکه‌های مدار مجازی، آنست که در حین تنظیم یک مدار مجازی، بین ماشین میزبان و زیرشبکه توافقاتی صورت بگیرد. در این توافقات، عموماً حجم و شکل ترافیک، کیفیت مورد نیاز خدمات و پارامترهای دیگر مشخص می‌شوند. به منظور عمل به مفاد این قرارداد، زیرشبکه در هنگام تنظیم مسیر، منابع مورد نیاز را در طول مسیر، از قبل کنار می‌گذارد. این منابع شامل فضای بافر، فضای جدول در هر مسیریاب و پهنای باند خطوط است. در این روش احتمال بروز ازدحام در مدارات مجازی نادر است چرا که تمام منابع مورد نیاز از قبل رزرو و موجود بودن آن تضمین می‌شود.

این گونه رزروسازی می‌تواند به صورت یک روال عملیاتی استاندارد همیشه انجام شود و یا فقط در شرایطی اتخاذ شود که شبکه با ازدحام مواجه شده است. اشکال استفادهٔ همیشگی از این روش آن است که می‌تواند منابع را هدر بدهد. به عنوان مثال اگر شش مدار مجازی که هر یک به پهنای باند 1Mbps نیازمندند همگی از یک خط فیزیکی 6Mbps عبور کرده باشند مسیریاب مجبور به علامتگذاری آن به عنوان خط پر خواهد بود در حالی که به ندرت اتفاق می‌افتد که هر شش مدار مجازی بطور همزمان مورد استفاده قرار بگیرند. نتیجتاً در شرایط معمولی هزینه‌ای که برای کنترل ازدحام پرداخت می‌شود پهنای باند بلااستفاده (تلفاتی) است.

۵-۳-۴ کنترل ازدحام در زیرشبکه‌های دیتاگرام

حال اجازه بدهید به راهکارهایی بپردازیم که می‌توان از آنها در زیرشبکه‌های دیتاگرام (و همچنین زیرشبکه‌های مدار مجازی) بهره گرفت. هر مسیریاب می‌تواند براحتی بر میزان بهره‌وری (Utilization) خطوط خروجی و دیگر منابع خودش نظارت داشته باشد. به عنوان مثال مسیریاب می‌تواند به هر یک از خطوط خود یک متغیر اعشاری u که مقداری بین 0.0 تا 1.0 دارد نسبت بدهد. مقدار این متغیر میزان بهره‌وری خط را منعکس می‌کند. برای آنکه بتوان تخمین دقیقی از u داشت می‌توان بطور متناوب و در لحظات خاصّ از میزان بهره‌وری خط[۱] نمونه‌برداری کرد و با فرض آنکه این مقدار f محاسبه شده باشد (f نیز بین صفر و یک است)، مقدار u را طبق رابطهٔ زیر بهنگام سازی نمود:

$$u_{new} = \propto . u_{old} + (1 - \propto) . F$$

∝ یک ثابت است که تعیین می‌کند مسیریاب با چه سرعتی وضعیت گذشته را فراموش خواهد کرد.

هر گاه u از یک مقدار آستانه (Threshold) [یا به عبارتی از حدّ مجاز] تجاوز کند آن خط خروجی در وضعیت «هشدار» وارد می‌شود و در این صورت باید عملیاتی برای خروج از این وضعیت انجام گیرد. این عملیات می‌تواند یکی از گزینه‌های زیر باشد که در ادامه آنها را تشریح خواهیم کرد.

بیت هشدار (The Warning Bit)

در معماری قدیم شبکه‌های DECNET ، «وضعیت هشدار» با تنظیم یک بیت خاص در سرآیند بسته‌ها، اعلام می‌شد. در شبکهٔ Frame Relay نیز همینگونه است. وقتی بسته‌ای به مقصد خود می‌رسید، لایهٔ انتقال همان بیت را در درون بستهٔ Ack کپی کرده و آن را برای مبداء پس می‌فرستاد و بدین نحو ماشین مبداء ترافیک خود را تقلیل می‌داد.

مادامیکه مسیریاب در وضعیت هشدار قرار داشت، تنظیم این بیت نیز ادامه می‌یافت بدین معنا که ماشین مبداء در بسته‌های بعدی Ack بازهم این بیت را دریافت می‌کرد. مبداء نیز با شمارش بسته‌های Ack حساب می‌کرد که در چه کسری از این بسته‌ها بیت هشدار وجود دارد و براساس آن نرخ انتقال خود را تنظیم می‌نمود. تا وقتی که دریافت این بیتها ادامه می‌یافت مبداء نیز به کاهش نرخ ارسال خود ادامه می‌داد. پس از کاهش نرخ ارسال تا حدّ نهایی، مجدداً نرخ ارسال شروع به افزایش می‌کرد [در صورت خروج از وضعیت هشدار]. دقت کنید که چون هر مسیریاب واقع بر روی مسیر می‌توانست «بیت هشدار» را به ۱ تنظیم کند لذا فقط زمانی ترافیک افزایش می‌یافت که هیچ یک از مسیریابها مشکلی نداشتند.

بسته‌های دعوت به آرامش (Choke Packet)

الگوریتم کنترل ازدحام قبلی نسبتاً زیرکانه است چراکه با یک بیان غیرمستقیم و تلویحی به مبداء تفهیم می‌کند که باید از سرعت خود بکاهد. چرا این کار مستقیماً انجام نشود؟ برای این کار مسیریاب یک بسته به نام **«دعوت به آرامش»** (Choke Packet) به ماشین مبداء بر گردانده و در آن آدرس مقصد بسته را نیز درج می‌کند [تا ماشین مبداء آگاه شود بسته‌های ارسالی او در راه رسیدن به کدام مقصد در مسیر پر ازدحام قرار گرفته‌اند و بداند که نرخ ارسال برای چه مقصدی را باید کاهش بدهد چرا که یک ماشین ممکن است بطور همزمان برای چند ماشین بسته ارسال کند -م]. البته بستهٔ اصلی، علامتگذاری شده و به راه خود ادامه می‌دهد (فقط یک بیت خاص در سرآیند

---

۱. نسبت داده‌های ارسالی بر روی خطّ به ظرفیت کلّ خط را بهره‌وری آن خطّ در نظر بگیرید. اندازه‌گیری این نسبت برای هر خط چندان دشوار نیست و برخی از سخت‌افزارهای شبکه امکان این اندازه‌گیری را در اختیار می‌گذارند. -م

بسته تنظیم می‌شود). این علامتگذاری از آن جهت انجام می‌شود که بسته‌های «دعوت به آرامش» بیشتری در طول مسیر تولید نشود؛ سپس بستهٔ اصلی بروش معمول به سوی مقصد خود هدایت می‌شود.

هر گاه ماشین مبداء، بستهٔ دعوت به آرامش دریافت کند، ملزم به کاهش ترافیک ارسالی بدان مقصد خاصّ (تا X درصد) می‌باشد. از آنجایی که ممکن است بسته‌های دیگری که قبلاً به همان مقصد روانه شده‌اند در همان مسیر حرکت کرده باشند و آنها نیز منجر به تولید بسته‌های «دعوت به آرامش» دیگری شده باشند لذا ماشین مبداء پس از دریافت اولین بستهٔ «دعوت به آرامش» به مدت زمان ثابت و مشخصی بسته‌هایی از این نوع را نادیده می‌گیرد. پس از طی این مدت، ماشین میزبان مجدداً به اندازهٔ زمان مشخصی گوش می‌دهد که ببیند آیا «بستهٔ دعوت به آرامش» دیگری دریافت می‌شود؟ اگر چنین بسته‌ای دریافت شد، خط کماکان در وضعیت ازدحام است فلذا باز هم جریان داده‌های خود را کاهش می‌دهد و مجدداً برای مدتی بسته‌های دعوت به آرامش را نادیده می‌گیرد. برعکس اگر در خلال مدت گوش دادن هیچ بستهٔ دعوت به آرامش دریافت نشد، ماشین میزبان می‌تواند جریان داده‌های خود را افزایش بدهد. در این پروتکل «فیدبک ضمنی» کمک می‌کند تا بتوان قبل از آنکه جریان بسته‌ها بطور کلّ مسدود شود از ازدحام پیشگیری کرد مگر آنکه مشکلی جدی رخ بدهد.

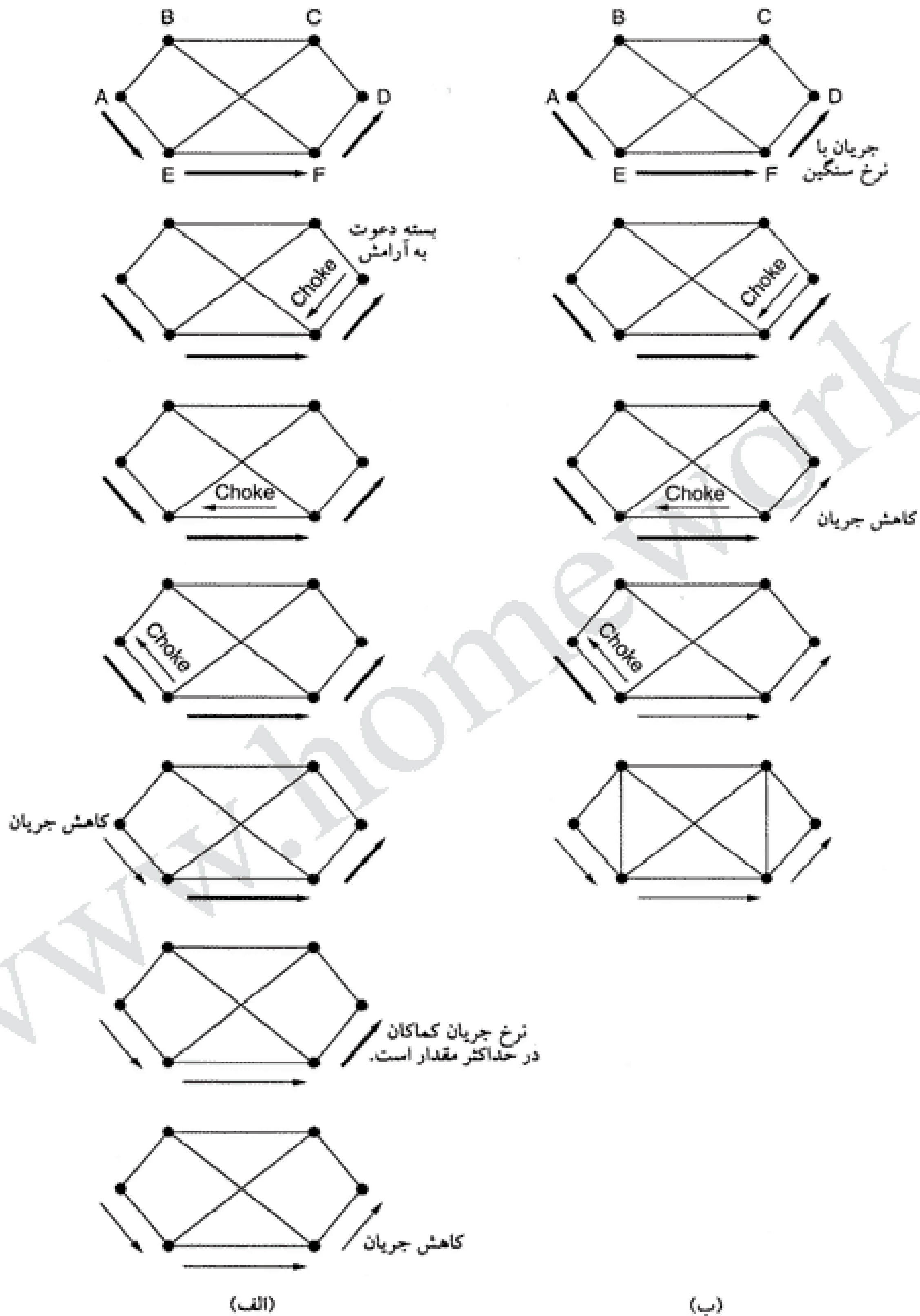
ماشینهای میزبان می‌توانند با تنظیم پارامترها و تغییر در سیاستهای کنترل ازدحام مثل اندازهٔ پنجره، ترافیک ارسالی خود را کاهش بدهند. عموماً دریافت یک بستهٔ «دعوت به آرامش» باعث پنجاه درصد کاهش در نرخ قبلی ارسال می‌شود. بستهٔ بعدی، ترافیک را به ۲۵ درصد تقلیل می‌دهد و به همین ترتیب. [به عبارت دیگر دریافت بستهٔ دعوت به آرامش مقدار فعلی نرخ ترافیک را عموماً نصف می‌کند. -م] افزایش نرخ ارسال معمولاً ضریب رشد کمتری دارد تا از بروز مجدد و پیاپی ازدحام پیشگیری شود. [یعنی مثلاً اگر دریافت بستهٔ دعوت به آرامش نرخ ارسال را نصف می‌کند عدم دریافت مجدد آن در یک دورهٔ زمانی مشخص، باعث دو برابر شدن نرخ ارسال نخواهد شد. -م]

چندین گونه از این الگوریتم برای کنترل ازدحام ارائه شده است. مثلاً در یکی از آنها، مسیریاب چندین سطح آستانه (Threshold) در نظر می‌گیرد. بسته به آنکه وضعیت از کدامیک از سطوح تجاوز کرده باشد بسته‌های دعوت به آرامش می‌توانند در برگیرند؛ یکی از سه هشدار «وضعیت احتیاط»، «وضعیت حادّ» و «بحران قطعی» باشند.

در گونهٔ دیگری از این الگوریتم برای صدور علائم هشداردهنده، بجای استفاده از معیار بهره‌وری خط از معیار طول صف یا میزان استفاده از فضای بافر بهره گرفته می‌شود. البته می‌توان از یک رابطهٔ وزن‌دهی نمایی (Exponential Weighting) استفاده کرد و براساس تمامی این معیارها یک مقدار تلفیقی برای میزان بهره‌وری (u) تعریف نمود.

بسته‌های دعوت به آرامش گام‌به گام (Hop-by-Hop Choke Packet)

در سرعتهای بالا یا در مسیرهای طولانی، ارسال بسته‌های «دعوت به آرامش» به ماشین مبداء، کارآیی خوبی نخواهد داشت چراکه واکنش این روش بسیار کُند است. به عنوان مثال فرض کنید که ماشینی در سان‌فرانسیسکو (مثلاً مسیریاب A در شکل ۵-۲۸) ترافیکی از بسته‌ها را با سرعت ۱۵۵ مگابیت در ثانیه برای ماشینی در نیویورک (مثلاً مسیریاب D در شکل ۵-۲۸) ارسال کند. اگر بافر ماشین در نیویورک، رو به پرشدن بگذارد حدوداً سی میلی‌ثانیه طول می‌کشد تا بسته‌های دعوت به آرامش به سان‌فرانسیسکو برگردد و از مبداء بخواهد کُندتر ارسال کند. در شکل ۵-۲۸-الف انتشار و بازگشت «بستهٔ دعوت به آرامش» در مراحل دوم، سوم و چهارم نشان داده شده است. در خلال این سی میلی‌ثانیه تاخیر، ۴/۶ مگابیت اطلاعات دیگر فرستاده خواهد شد. حتی اگر پس از دریافت



شکل ۵-۲۸. (الف) یک بستهٔ دعوت به آرامش که فقط مبداء را تحت تاثیر قرار می‌دهد. (ب) یک بستهٔ دعوت به آرامش که در هر گام، بر روی مسیریابها واقع بر مسیر تاثیر می‌گذارد.

این هشدار ماشین واقع در سانفرانسیسکو فوراً ارسال خود را بطور کلّ قطع نماید، ۴/۶ مگابیت اطلاعات بر روی خط به سوی ماشین روانه هستند و باید به نحوی تکلیف آنها مشخص شود. فقط در مرحلهٔ هفتم از شکل ۵-۲۸-الف مسیریاب واقع در نیویورک متوجه کاهش ترافیک خواهد شد.

راهکار دیگر آن است که بستهٔ دعوت به آرامش در هر گام از مسیری که می‌پیماید تأثیری بر عملکرد مسیریابهای میانی بگذارد. (به شکل ۵-۲۸-ب دقت کنید). در اینجا به محض آنکه بستهٔ دعوت به آرامش به مسیریاب F می‌رسد، F ملزم به کاهش ترافیک ارسالی به D خواهد بود. انجام این کار منوط به آن است که F فضای بافر بیشتری را برای جریان بسته‌ها اختصاص بدهد چراکه ماشین مبداء با سرعت کامل در حال ارسال است و F در این میان می‌تواند کمکی موقت و سریع به D بنماید [با ایجاد توقف مصنوعی]؛ این کار را می‌توانید همانند آگهی‌های تجاری فرض کنید که شما را موقتاً از سردرد ناشی از دیدن یک برنامهٔ طولانی نجات می‌دهند!! در گام بعدی بستهٔ دعوت به آرامش به E می‌رسد و به او نیز تفهیم می‌شود که ترافیک خود به F را کاهش بدهد. این عمل نیز مستلزم وجود فضای بافر بیشتری در E است ولی F را موقتاً یاری می‌دهد. نهایتاً بستهٔ دعوت به آرامش به A می‌رسد و جریان بطور واقعی و از مبداء آن کاهش می‌یابد.

تأثیر نهایی این روش گام‌به‌گام (Hop-by-Hop) آن است که در لحظه بروز ازدحام به سرعت چاره‌ای اندیشیده می‌شود ولی این راهکار به بهای افزایش فضای بافر ارسال، تمام خواهد شد. در این روش می‌توان بدون از دست رفتن هیچ بسته‌ای، مانع از افزایش ازدحام شد. این نظریه و نتایج شبیه‌سازی آن در مرجع (Mishra and Kanakia, 1992) تشریح شده است.

۵-۳-۵　دور ریختن بار (Load Shedding)

هر گاه هیچیک از روشهای فوق مشکل ازدحام را رفع نکند، مسیریابها می‌توانند آخرین تیر ترکش خود را بیازمایند: «**دور ریختن بار**»! عمل دور ریختن بار، آخرین و بدترین روشی است که به مسیریاب اجازه می‌دهد هرگاه با سیل بسته‌هایی که نمی‌تواند از عهدهٔ هدایت آنها برآید، مواجه شود آنها را دور بریزد. (این اصطلاح از ادبیات شرکتهای تولید انرژی برق گرفته شده که در آنجا نیز مثلاً در یک روز گرم تابستان که مصرف برق از میزان تولید بیشتر می‌شود، برق بخشی از مناطق عمداً قطع می‌شود تا کل مناطق با مشکل قطع برق مواجه نشوند!)

یک مسیریاب که غرق در بسته‌های اطلاعاتی شده می‌تواند تصادفاً برخی از آنها را انتخاب و حذف کند ولی از این بهتر هم می‌تواند عمل کند. اینکه کدام بسته باید حذف شود به نوع برنامهٔ کاربردی که آنرا تولید کرده بستگی دارد. در انتقال فایل بسته‌های قدیمی ارزشمندتر از بسته‌های جدید هستند چراکه حذف بستهٔ ۶ و نگه داشتن بسته‌های ۷ تا ۱۰ باعث ایجاد یک شکاف در میان داده‌های گیرنده شده و ممکن است فرستنده مجبور شود بسته‌های ۶ تا ۱۰ را نیز از نو ارسال کند (اگر گیرنده بطور طبیعی بسته‌های خارج از ترتیب را حذف کند.) در یک فایل ۱۲ بسته‌ای، حذف بستهٔ ۶ ممکن است منجر به تکرار ارسال بسته‌های ۶ تا ۱۰ شود در حالی که حذف بستهٔ ۱۰ ممکن است فقط به تکرار بسته‌های ۱۰ تا ۱۲ نیاز داشته باشد. در طرف مقابل برای کاربردهای چند رسانه‌ای (مثل ارسال صدا یا تصویر)، بسته‌های جدید مهمتر از بسته‌های قدیمند.

برای آن که سطح هوشمندی الگوریتم حذف بسته‌ها از این هم فراتر برود به همکاری فرستندهٔ بسته‌ها نیاز است. در بسیاری از برنامه‌های کاربردی، برخی از بسته‌ها بسیار مهمتر از بقیه هستند. به عنوان مثال در برخی الگوریتمهای ارسال ویدیوی فشرده شده، در لحظات خاصی یک فریم تصویر کامل ارسال می‌شود و پس از آن فریمهای تصویر بعدی کامل نیستند بلکه فقط تفاوت این فریمها با آخرین فریم کامل محاسبه و پس از فشرده‌سازی ارسال می‌شوند. در چنین حالتی حذف بسته‌ای که بخشی از فریمهای فرعی محسوب می‌شود ارجح‌تر از حذف بسته‌ای است که به فریم اصلی و کامل تعلق دارد. به عنوان مثالی دیگر، انتقال یک سند

(Document) حاوی تصویر و متن ASCII را در نظر بگیرید. از دست دادن یک خط از نقاط تصویر در برخی از عکسها، کم ضررتر از دست دادن یک خط از متن است.

برای پیاده‌سازی یک سیاست هوشمند برای حذف بسته‌ها، برنامه‌های کاربردی باید ردهٔ اولویت مورد نظر را در بسته‌های خود مشخص کنند تا میزان اهمیت آنها مشخص گردد. اگر چنین کاری انجام شده باشد زمانی که مسیریاب مجبور به حذف بسته‌ها می‌شود می‌تواند از بسته‌های با اولویت پایین آغاز کند و بعداً به سراغ رده‌های بالاتر برود. البته فقط در موارد خاصّ می‌توان بسته‌هایی را در بالاترین ردهٔ اولویت علامتگذاری کرد.

البته چون هر ماشین در ارسال بسته‌ها با هر اولویتی، آزادی عمل دارد لذا باید به نحوی در کاربر انگیزه ایجاد کرد تا بسته‌هایش را با اولویت پایین بفرستد؛ **پول** می‌تواند این انگیزه را ایجاد کند؛ اگر برای حمل بسته‌های با اولویت بالا هزینهٔ بیشتری گرفته شود کاربران سعی می‌کنند بی‌مورد از اولویتهای بالا استفاده نکنند. البته فرستنده ممکن است اجازهٔ ارسال بسته‌های با اولویت بالا را در شرایط بار سبک به کاربر بدهد ولیکن با افزایش بار حذف خواهند شد. این موضوع کاربران را تشویق می‌کند از ارسال بی‌مورد بسته‌های با اولویت بالا صرفنظر کنند.

گزینهٔ دیگر آنست که به ماشینهای میزبان اجازه بدهیم گاهی بیش از حد توافق شده در هنگام ایجاد مدار مجازی، ارسال داشته باشند ولیکن مشروط بدانکه ترافیک اضافی اولویت پایینی داشته باشد و به محض آشکار شدن علائم ازدحام حذف شود. چنین روشی ایدهٔ بدی نیست چراکه از منابع آزاد سیستم استفادهٔ مفید می‌شود: به ماشینهای میزبان اجازه داده‌ایم مادامی که کس دیگر به منابع آزاد نیاز نداشته باشد از آن استفاده کنند ولیکن در شرایط دشوار [بار بالا] حقی برای کسی ایجاد نکرده‌ایم.

تشخیص زودهنگام (Random Early Detection)

بر هر کسی روشن است که رفع ازدحام به محض کشف علائم آن بسیار کارآمدتر از آن است که بگذاریم اوضاع را بهم بریزد و سپس به رفع آن اقدام کنیم. چنین تجربه‌ای بدین ایده منتهی می‌شود که قبل از اشباع شدن کل فضای بافر، مسیریاب به حذف برخی از بسته‌ها اقدام کند. الگوریتم رایجی که در این خصوص کاربرد دارد اصطلاحاً الگوریتم RED (Random Early Detection) نامیده می‌شود. در برخی از پروتکلهای لایهٔ انتقال (مثل TCP) اگر بسته‌ها در حین انتقال از بین بروند مبداء، نرخ ارسال بسته‌هایش را کاهش می‌دهد. استدلالی که در پشت این منطق نهفته است آن است که TCP در اصل برای شبکه‌های سیمی طراحی شده و از آنجایی که شبکه‌های سیمی بسیار قابل‌اعتماد و کم خطا هستند لذا دلیل از دست رفتن بسته‌ها اغلب ناشی از پر شدن بافر است تا خطاهای انتقال. از این حقیقت می‌توان برای کمک به کاهش ازدحام بهره گرفت.

دلیل آنکه اجازه می‌دهیم مسیریاب قبل از آنکه وضعیت وخیم شود بسته‌ها را حذف کند آن است که بتوان قبل از دیر شدن کاری انجام داد. برای تعیین زمان شروع حذف بسته‌ها، هر مسیریاب میانگین طول صفهای خود را نگه می‌دارد. هر گاه طول متوسط صف بر روی برخی از خطوط، از حد مجاز تجاوز کرد آن خط با ازدحام مواجه شده و عملیات حذف شروع می‌شود.

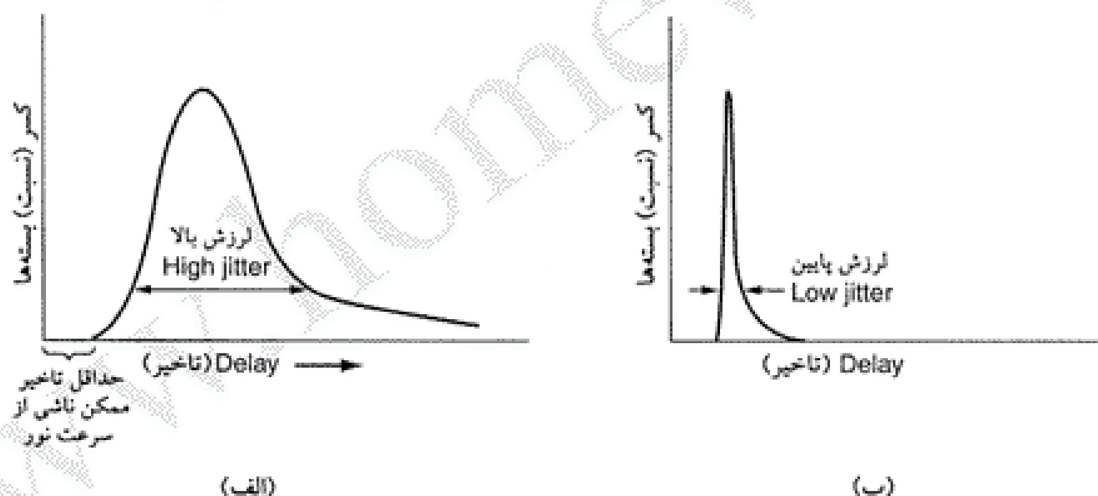
از آنجا که مسیریاب احتمالاً نمی‌تواند بفهمد کدام مبداء باعث مشکل بیشتری شده لذا این ایده که یکی از بسته‌ها را تصادفاً از صف دچار ازدحام بیرون بکشد، می‌تواند ایدهٔ خوبی باشد.

مسیریاب چگونه می‌تواند بروز مشکل را به مبداء بسته اعلام کند؟ یک روش آن است که یک «بستهٔ دعوت به آرامش» برای مبداء بسته ارسال شود. اشکال این روش در آن است که بار بیشتری را بر روی شبکه‌ای که از قبل دچار ازدحام شده، تحمیل می‌کند. راهکار دیگر آن است که بستهٔ انتخاب شده را حذف کرده و هیچ گزارشی هم اعلام نکند. در این صورت مبداء بسته از عدم برگشت بستهٔ اعلام وصول (Ack) متوجه حذف بسته شده و اقدام

لازم را صورت می‌دهد. از آنجایی که مبداء بسته می‌داند که از بین رفتن بسته‌ها ناشی از ازدحام و حذف بسته‌هاست لذا بجای آنکه سخت‌تر تلاش کند، سرعت خود را کاهش می‌دهد. این «فیدبک ضمنی» صرفاً زمانی کار می‌کند که مبداء بسته وقتی متوجه از دست رفتن آن شود سرعت خود را کاهش بدهد. در شبکه‌های بی‌سیم که از بین رفتن بسته‌ها بیشتر ناشی از نویز لینک رادیویی است نمی‌توان از این روش استفاده کرد.

۶-۳-۵ کنترل لرزش (Jitter Control)

برای کاربردهایی مثل ارسال جریان داده‌های صدا و تصویر (ویدئو)، این که آیا ۲۰ میلی‌ثانیه طول می‌کشد تا بسته‌ای تحویل مقصد شود یا ۳۰ میلی‌ثانیه، موضوع مهمی نیست بلکه مهم آنست که این زمان انتقال ثابت باشد. میزان تغییر در زمان رسیدن بسته‌ها (یا به عبارتی انحراف معیار زمان تحویل بسته‌ها) اصطلاحاً **«لرزش»** (Jitter) نامیده می‌شود. وقتی «میزان لرزش» بالاست یعنی مثلاً وقتی که برخی از بسته‌ها در ۲۰ میلی‌ثانیه و برخی دیگر در ۳۰ میلی‌ثانیه به مقصد می‌رسند، کیفیت ارسال صدا و تصویر ویدیوئی را کاهش می‌دهد. نمودار «لرزش» در شکل ۵-۲۹ به تصویر کشیده شده است. در این مثال، هر گاه ۹۹ درصد از بسته‌ها تأخیری بین ۲۴/۵ تا ۲۵/۵ میلی‌ثانیه داشته باشند، برای ارسال صدا و تصویر احتمالاً مناسب و قابل قبول خواهد بود. البته باید بازهٔ تغییرات تأخیر، قابل تحقق و امکان‌پذیر باشد. در ضمن باید تأخیر ناشی از انتقال سیگنال با سرعت نور و همچنین حداقل تأخیر عبور از مسیریابها را به حساب آورد ولیکن می‌توان برخی از تأخیرهای کوچک و غیرقابل پیش‌بینی را نادیده گرفت.



شکل ۵-۲۹. (الف) لرزش زیاد (ب) لرزش کم.

«لرزش» را می‌توان با محاسبهٔ میانگین زمان انتقال در هر گام از کلّ مسیر، محدود کرد: وقتی بسته‌ای به یک مسیریاب می‌رسد آن مسیریاب بررسی می‌کند تا ببیند آن بسته به چه مقدار از برنامهٔ زمانی خود جلوتر یا عقب‌تر است. این اطلاعات در درون هر بسته ذخیره شده و در هر گام بهنگام می‌شود. اگر بسته از برنامهٔ زمانی خود جلو باشد آنقدر معطل می‌شود تا به برنامهٔ زمانی خود برگردد ولیکن اگر از برنامه عقب افتاده باشد مسیریاب سعی می‌کند آنرا به سرعت بر روی خروجی بفرستد.

در حقیقت الگوریتمی که تعیین می‌کند کدامیک از بسته‌های منتظر ارسال، برای خروج از یک خط اولویت دارد می‌تواند بسته‌ای را انتخاب کند که بیشتر از بقیه، از برنامهٔ زمانی خود عقب‌افتاده است. بدین ترتیب بسته‌هایی که از برنامهٔ زمانی خود جلوتر هستند آهسته‌تر و بسته‌هایی که از برنامهٔ زمانی خود عقب افتاده‌اند سریعتر هدایت می‌شوند و در هر دو حالت میزان لرزش کاهش خواهد یافت.

در برخی از کاربردها مثل ارسال «ویدئو برحسب تقاضا» (Video on-Demand)، خود گیرنده می‌تواند با بافر

کردن بسته‌ها و استخراج آنها از بافر جهت نمایش [در زمان مناسب] میزان لرزش را کاهش بدهد (بجای آنکه این کار به صورت بی‌درنگ به شبکه محول شود). ولیکن در برخی دیگر از کاربردها خصوصاً زمانی که نیاز به محاورهٔ بی‌درنگ بین مردم باشد (مثل تلفن اینترنتی یا کنفرانسهای ویدیویی از راه دور)، تأخیر ناشی از بافر کردن داده‌ها، قابل قبول نخواهد بود.

روشهای کنترل ازدحام، حوزه‌ای فعّال برای پژوهش و تحقیق است. پژوهشهای تازه و دستاوردهایی که در این زمینه حاصل شده در مرجع (Gevros et al., 2001) جمع‌بندی و ارائه شده است.

۴-۵ کیفیت خدمات (Quality of Service)

تکنیکهایی که در بخشهای قبلی مرور کردیم به منظور کاهش ازدحام و بهبود کارآیی شبکه طراحی شده بودند. با این حال با رشد شبکه‌های چندرسانه‌ای، این ارزیابی‌های خاصّ و تمهیدات موردی کفایت نمی‌کند و به تلاشی جدّی در تضمین کیفیت خدمات شبکه و طراحی پروتکل ویژه نیاز است. در بخشهای آتی نیز مطالعات خود را در خصوص کارآیی شبکه، ادامه خواهیم داد با این تفاوت که تمرکز ویژه‌ای بر روی روشهای «تضمین کیفیت خدمات» متناسب با نیاز برنامه‌های کاربردی، خواهیم داشت. ولی در همین ابتدا باید متذکر شویم که بسیاری از این ایده‌ها در حال تغییر هستند و عوض خواهند شد.

۱-۴-۵ نیازها

به دنباله‌ای از بسته‌ها که از مبداء به سوی مقصد روانه می‌شوند اصطلاحاً **«جریان»** (Flow) گفته می‌شود. درشبکه‌های اتصال‌گرا تمام بسته‌هایی که به یک «جریان» تعلق دارند، مسیر مشابهی را طی می‌کنند در حالی که در شبکه‌های بدون‌اتصال (Connectionless) بسته‌ها ممکن است از مسیرهای متفاوتی حرکت کنند. نیازهای مطلوب برای هر «جریان» را می‌توان با چهار پارامتر ابتدایی مشخص کرد: (۱) **قابلیت اطمینان** (۲) **تأخیر** (۳) **لرزش** (Jitter) (۴) **پهنای باند**. مجموعهٔ این چهار پارامتر، «کیفیت خدمات» یا به اختصار QoS مورد نیاز یک «جریان» را تعیین می‌کند. در شکل ۵-۳۰ برخی از کاربردهای رایج و سطح نیازمندیهای آنها فهرست شده‌اند.

نوع کاربرد	قابلیت اطمینان	تاخیر	لرزش	پهنای باند
پست الکترونیکی	بالا	پایین	پایین	پایین
انتقال فایل	بالا	پایین	پایین	متوسط
دسترسی به وب	بالا	متوسط	پایین	متوسط
ورود به سیستم از راه دور	بالا	متوسط	متوسط	پایین
دریافت صوت برحسب تقاضا	پایین	پایین	بالا	متوسط
دریافت ویدیو برحسب تقاضا	پایین	پایین	بالا	بالا
تلفن اینترنتی	پایین	بالا	بالا	پایین
ویدیو کنفرانس	پایین	بالا	بالا	بالا

شکل ۵-۳۰. سطح نیازمندی برنامه‌های کاربردی به کیفیت خدمات.

چهار کاربرد اول نیاز شدیدی به «قابلیت اطمینان» (Reliability) دارند یعنی هیچ یک از بیتها نباید اشتباه تحویل شوند. [به عبارت دیگر حتی یک بیت خطا در کل جریان داده‌ها قابل قبول نیست.] با اضافه کردن کدهای کشف خطا به هر بسته و بررسی آنها در مقصد می‌توان به این هدف نائل شد. اگر بسته‌ای در حین انتقال آسیب دید، وصول آن اعلام و تصدیق نمی‌شود و طبعاً باید از نو ارسال شود. این استراتژی قابلیت اطمینان بالایی برای بسته‌ها

فراهم می‌کند. چهار کاربرد آخر (صدا / ویدیو) می‌توانند اندکی خطا را تحمل کنند لذا کدهای کشف خطا برای هر بسته محاسبه و بررسی نخواهد شد.

کاربردهای انتقال فایل، شامل پست الکترونیکی یا دریافت تصاویر، حساسیت چندانی به تأخیر ندارند. حتی اگر تمام بسته‌ها به طور یکنواخت تا چند ثانیه هم تأخیر داشته باشند هیچ مشکل حادی پدید نمی‌آید. کاربردهای محاوره‌ای (Interactive) همانند جستجو در وب یا ورود به سیستم از راه دور (Remote Login) حساسیت بیشتری به تأخیر دارند.

کاربردهای بی‌درنگ مثل تلفن یا کنفرانس از راه دور حساسیت شدیدی به تأخیر دارند. اگر کلمات ادا شده در یک تماس تلفنی همگی با 2.000 ثانیه تأخیر برسند. کاربران چنین تماسی را نامناسب و غیرقابل قبول خواهند داشت. برعکس، اجرای فایلهای صدا یا ویدیو که بر روی یک سرویس‌دهنده ذخیره شده به تأخیر پایین نیاز ندارند.[۱]

سه کاربرد اول حساسیت چندانی به رسیدن بسته‌ها با فاصلهٔ زمانی نامشخص (نسبت به یکدیگر) ندارند؛ به عبارت بهتر لرزش بالا (High Jitter) مشکل چندانی برای این کاربردها ایجاد نمی‌کند. کاربرد چهارم یعنی ورود به سیستم از راه دور (Remote Login) تا حدودی به لرزش حساستر هستند چراکه در اثر لرزش زیاد ممکن است کاراکترها به صورت ناگهانی و نامتعارف بر روی صفحه نمایش ظاهر شوند. تصاویر ویدیویی و خصوصاً صدا شدیداً به «لرزش» حساس هستند. همانگونه که اشاره شد کاربری که در حال تماشای یک نمایش ویدیویی غیرزنده از روی شبکه است و فریمهای تصویر، همگی با 2.000 ثانیه تأخیر می‌رسند مشکلی پدید نخواهد آمد در حالی که اگر زمان انتقال بین ۱ تا ۲ ثانیه متغیر باشد، مشکلات جدی ایجاد خواهد شد. برای «صدا» لرزش چند میلی‌ثانیه‌ای نیز بوضوح شنیده خواهد شد.

در آخر، کاربردها نیاز متفاوتی به پهنای باند دارند: پست الکترونیکی و Remote Login به پهنای باند زیادی نیاز ندارند در حالی که ارسال ویدیو در تمام أشکال [فشرده شده یا غیرفشرده] پهنای باند بسیار زیادی را می‌طلبد.

شبکه‌های ATM ، «جریان» (Flow) را براساس QoS مورد نیاز در چهار ردهٔ وسیع دسته‌بندی کرده است:

۱. ارسال با نرخ ثابت (مثل تلفن از طریق شبکه)
۲. ارسال بی‌درنگ با نرخ متغیر (مثل کنفرانس ویدیویی فشرده شده)
۳. ارسال غیربی‌درنگ با نرخ متغیر (مثل تماشای نمایش غیرزنده از طریق اینترنت)
۴. ارسال با هر نرخ ممکن (برای انتقال فایل) (Available Bit Rate)

این رده‌ها در شبکه‌های دیگر و اهداف دیگر نیز مفید و راهگشا هستند. ردهٔ «ارسال با نرخ ثابت» تلاشی است در جهت شبیه‌سازی یک اتصال سیمی با پهنای باند ثابت و تاخیر یکنواخت.

«ارسال با نرخ متغیر» زمانی مفید خواهد بود که تصاویر ویدیویی، فشرده شده باشند و ضریب فشرده سازی فریمهای تصویر با هم فرق کند. بنابراین ارسال یک فریم تصویر با جزئیات و ظرافت زیاد نیاز به ارسال تعداد بیت زیادی دارد در حالیکه ارسال یک فریم تصویر که از یک دیوار سفید گرفته شده ممکن است تا حد بسیار بالایی فشرده شود. ردهٔ «ارسال با هر نرخ ممکن» برای کاربردهایی مثل پست الکترونیکی کاربرد دارد که به تأخیر یا لرزش حساس نیستند.

۵-۴-۲ راهکارهای دستیابی به کیفیت خوب خدمات

تا اینجا چیزهایی را در خصوص نیازهای QoS آموخته‌ایم؛ سؤال این است که چگونه به آنها برسیم؟ در همین جا

---

۱. چراکه مشاهدهٔ یک فیلم یا شنیدن یک موسیقی غیرزنده با چند ثانیه تأخیر دائم مشکل حادی ایجاد نمی‌کند. -م

باید اذعان داشت که هیچ راهکار واحد و جادویی برای نیل به این اهداف وجود ندارد. به عبارت دیگر روشی واحد که تمام نیازهای QoS را به صورت بهینه برآورده نماید موجود نیست بلکه راهکارهای گوناگونی تعریف شده که در آنها برای ارائه راه حلهای عملی، ترکیبی از چندین روش مختلف بکار گرفته شده است. حال به بررسی برخی از این راهکارها که طراحان سیستم برای تأمین کیفیت خدمات از آنها بهره گرفته‌اند می‌پردازیم.

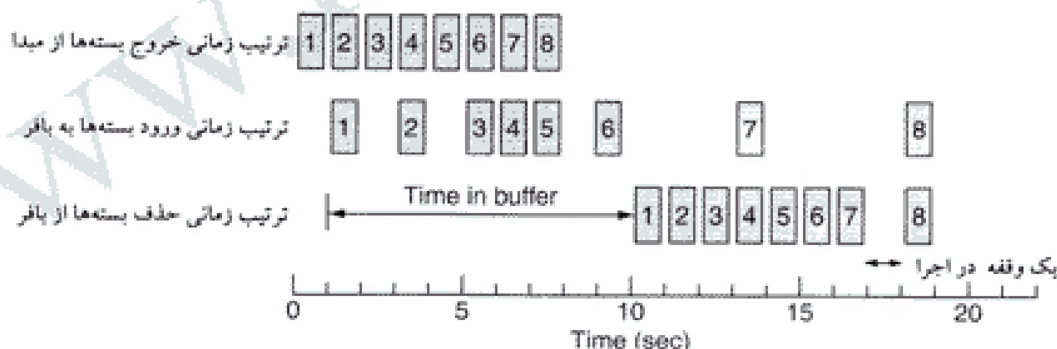
تمهیدات کافی

یک راه حل ساده آن است که برای مسیریاب ظرفیت پردازش، فضای بافر و پهنای باند زیادتری تدارک دیده شود تا بسته‌ها بتوانند بسادگی و بسرعت از آنها بگذرند. اشکال این راه حل آن است که گران تمام می‌شود! به مرور زمان، طراحان ایده‌های بهتری در خصوص پیش‌بینی میزان منابع مورد نیاز مطرح می‌کنند؛ در آن زمان شاید بتوان از این روش در عمل استفاده کرد. از این دیدگاه سیستم تلفن، سیستمی با تمهیدات کافی است و به ندرت گوشی تلفن را برمی‌دارید و فوراً بوق آزاد آنرا نمی‌شنوید زیرا ظرفیت این سیستم، بیش از حد معمولی تقاضا در نظر گرفته شده است.

بافر کردن

«جریان» داده‌ها می‌تواند در طرف گیرنده و قبل از تحویل به پروسهٔ اصلی، بافر شود. بافر کردن جریان تأثیری در قابلیت اطمینان ندارد، تأخیر را افزایش می‌دهد ولیکن در عوض میزان لرزش را متعادل‌تر می‌کند. برای «صدا و تصویر برحسب تقاضا»[1]، لرزش یک مشکل اساسی است و این روش کمک زیادی به حل آن می‌کند.

تفاوت بین لرزش زیاد و لرزش کم را در شکل ۵-۲۹ مشاهده کردیم. در شکل ۵-۳۱ دنباله‌ای از بسته‌ها را می‌بینید که با «لرزش» قابل توجهی تحویل می‌شوند. [فرض شده این بسته‌ها حاوی صدا یا تصویر هستند.] بستهٔ اول در لحظهٔ t=0 ارسال و در لحظهٔ t=1sec تحویل ماشین گیرنده شده است. بستهٔ دوم با تأخیر بیشتری مواجه شده و ۲ ثانیه در راه بوده تا برسد. به محض آنکه این بسته‌ها از راه می‌رسند در ماشین گیرنده بافر می‌شوند.



شکل ۵-۳۱. یکنواخت کردن استریم خروجی بکمک بافرسازی بسته‌ها.

در t=10sec اجرای آنها (Playback) شروع می‌شود. در این لحظه بسته‌های ۱ تا ۶ بافر شده‌اند فلذا می‌توان در فواصل زمانی مشخص و یکنواخت آنها را از بافر استخراج کرد و اجرای متعادل و مناسبی را انتظار داشت. متأسفانه لحظه‌ای که نوبت به اجرای بسته ۸ فرا می‌رسد به دلیل تأخیر نامتعارف، در بافر موجود نیست و اجرای آن باید تا زمان رسیدن آن متوقف شود که این مسئله یک توقف نامطلوب در اجرای موزیک یا فیلم ایجاد می‌کند. این مشکل را می‌توان با ایجاد تأخیر بیشتر در زمان شروع اجرا کاهش داد ولی در عوض به حجم بافر بزرگتری نیاز

---

۱. Video/Audio On-Damand

است. سایتهای وب تجاری که دارای صدا یا تصویر هستند از برنامه‌هایی برای اجرای صدا و تصویر استفاده می‌کنند که قبل از شروع اجرا، داده‌ها را تا ۱۰ ثانیه بافر می‌کنند.

شکل‌دهی ترافیک (Traffic Shaping)

در مثال بالا ماشین مبداء، بسته‌ها را در فواصل زمانی یکنواخت بیرون می‌فرستد، در حالی که مواردی وجود دارد که بسته‌ها بطور نامنظم تولید و ارسال می‌شوند و این مسئله ممکن است منجر به بروز ازدحام در شبکه شود. خروجی غیریکنواخت عموماً وقتی است که سرویس‌دهندهٔ مربوطه، بطور همزمان درگیر ارسال چندین «جریان ویدیویی یا صدا» است و در ضمن امکان عملیات دیگری مثل «جلو و عقب رفتن سریع» (Fast Forward/Rewind در اجرای صدا و تصویر) و همچنین احراز هویت کاربران و نظایر آنرا نیز فراهم آورده است. از طرفی در برخی از کاربردها مثل کنفرانس از راه دور، بافر کردن داده‌ها بطور کلّ منطقی نیست.[1] با این حال اگر بتوان کاری انجام داد تا سرویس‌دهنده (و کلاً ماشینهای میزبان) مجبور به ارسال با نرخ یکنواخت باشند، کیفیت خدمات (QoS) بهتر خواهد بود. در اینجا به بررسی روشی با عنوان «**شکل‌دهی به ترافیک**» می‌پردازیم که بجای آنکه بر ماشین گیرنده تمرکز داشته باشد، ترافیک تولیدی در سمت سرویس دهنده را متعادل و یکنواخت می‌کند.

«شکل‌دهی به ترافیک» در خصوص منظم کردن نرخ متوسط (و کاهش حالت انفجارگونه) تولید داده‌های ارسالی است. «پروتکل پنجرهٔ لغزان» (Sliding Window Protocol) که قبلاً مطالعه کردیم فقط میزان داده‌هایی را که بطور همزمان ارسال می‌شوند، محدود می‌کند ولی متوسط نرخ ارسال آنها را محدود نمی‌کند. در روش شکل‌دهی به ترافیک، وقتی یک «اتصال» تنظیم می‌شود، کاربر و زیرشبکه (یا به عبارتی مشتری و حامل) بر روی الگوی خاصی از ترافیک (شکل ارسال ترافیک) برای آن مدار توافق می‌کنند. گاهی به این توافق، اصطلاحاً «توافق بر روی سطح خدمات» (Service Level Agreement) گفته می‌شود. مادامیکه مشتری بر طبق این توافق عمل کند و بسته‌ها را براساس قرارداد مورد توافق ارسال نماید، حامل (یعنی زیرشبکه) نیز متعهد است که بسته‌ها را بطور منظم تحویل بدهد. این عمل ازدحام را کاهش داده و اجازه می‌دهد زیرشبکهٔ حامل بتواند برای عمل به تعهدات خود به فعالیت طبیعی ادامه بدهد. چنین توافقی برای عملیاتی مثل انتقال فایل چندان مهم نیست بلکه برای داده‌های بی‌درنگ مثل ارسال صدا و تصویر که شدیداً به کیفیت خدمات نیازمند است، اهمیت حیاتی دارد.

در نتیجه برای عملیات شکل‌دهی به جریان، مشتری به حامل می‌گوید: «الگوی ارسال من بدین نحو است؛ آیا از عهدهٔ آن بر می‌آیی؟» اگر زیرشبکه حامل این الگو را پذیرفت، مورد دیگری به میان خواهد آمد: حامل چگونه متوجه می‌شود که آیا مشتری از مفاد توافق پیروی می‌کند؟ و اگر پیروی نکرد چه عکس‌العملی باید نشان بدهد؟

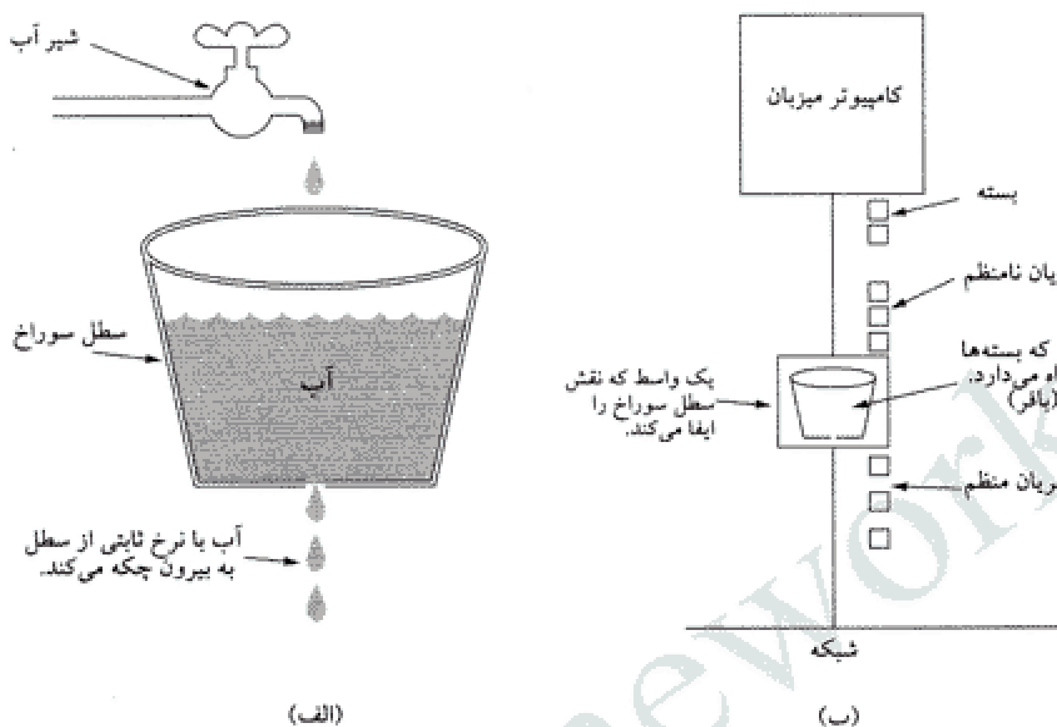
نظارت بر جریان ترافیک اصطلاحاً «**اعمال سیاست بر ترافیک**» (Traffic Policing) نامیده می‌شود. توافق در خصوص شکل ترافیک و نظارتهای بعدی، در زیرشبکه‌های مبتنی بر مدار مجازی ساده‌تر از زیرشبکه‌های مبتنی بر دیتاگرام است. با این حال حتی در زیرشبکه‌های دیتاگرام، از همین ایده‌ها می‌توان برای اتصالات ایجاد شده در لایهٔ انتقال استفاده کرد.

الگوریتم سطل سوراخ (Leaky Bucket Algorithm)

سطحی را در نظر بگیرید که همانند شکل ۵-۳۲-الف رخنهٔ کوچکی در کف آن وجود دارد. بدون توجه به نرخ ورود آب به این سطل، جریان نشتی از آن ثابت است: مادامیکه آب در سطل وجود دارد نرخ خروجی معادل ρ و

---

۱. بافر کردن مثلاً ۱۰ ثانیه از یک کنفرانس ویدیویی آنرا از حالت زنده خارج خواهد کرد و بهیچوجه مطلوب کاربران نیست. -م



شکل ۵-۳۲. (الف) یک سطل سوراخ‌دار آب (ب) یک سطل سوراخ‌دار «بسته».

وقتی سطل خالی است معادل صفر است. همچنین اگر آب اضافی به سطل پر وارد شود، از اطراف آن بیرون ریخته و هدر می‌رود. (یعنی در ظرفی دیگری که زیر این سوراخ قرار گرفته وارد نخواهد شد.)

به نحوی که در شکل ۵-۳۲-ب نشان داده شده از همین ایده می‌توان برای بسته‌ها نیز استفاده کرد. بطور مفهومی هر ماشین میزبان از طریق یک کارت واسط به شبکه متصل شده که این کارت واسط دارای سطلی سوراخ (یعنی یک صفّ داخلی بی‌نهایت) است. اگر بسته‌ای بخواهد در حالی که صفّ پر است وارد آن شود، حذف خواهد شد. به عبارت دیگر هرگاه یک یا چند پروسه در ماشین میزبان سعی در ارسال بسته‌ای نمایند در حالی که تعداد بسته‌های صفّ به حداکثر ممکن رسیده باشد، بستهٔ جدید بدون هیچ تشریفاتی حذف خواهد شد. این عملیات می‌تواند توسط سخت‌افزار واسط انجام شود یا آنکه توسط سیستم عامل ماشین میزبان شبیه‌سازی شود. این الگوریتم برای اولین بار توسط Turner (۱۹۸۶) پیشنهاد و الگوریتم سطل سوراخ نام گرفت. در حقیقت این روش چیزی نیست مگر یک سیستم صف‌بندی با یک سرویس‌دهندهٔ واحد که سرویس دهی به عناصر منتظر در صفّ با نرخ ثابتی انجام می‌شود.

ماشین میزبان اجازه دارد در هر **تیک ساعت** یک بسته بر روی شبکه قرار بدهد. این کار نیز می‌تواند توسط کارت واسط شبکه یا سیستم عامل ماشین انجام و مدیریت شود. این مکانیزم جریانی نامنظم و بی‌قاعده از بسته‌های تولید شده توسط پروسه‌های کاربری ماشین را به جریانی منظم بر روی شبکه تبدیل کرده و بدین ترتیب حالت انفجارگونهٔ ترافیک، معتدلتر شده و احتمال ازدحام کاهش چشمگیری می‌یابد.

اگر بسته‌ها دارای اندازه‌ای ثابت و یکسان باشند (مثل سلولهای ATM) می‌توان طبق توضیح بالا از این الگوریتم بهره گرفت: «**یک سلول در هر تیک ساعت**». ولیکن وقتی اندازهٔ بسته‌ها متغیر باشد، بهتر آن است که در هر تیک ساعت به جای یک بسته، تعداد بایت ثابتی ارسال شود. بدین ترتیب اگر قرار باشد در هر تیک ساعت ۱۰۲۴ بایت ارسال شود، می‌توان یک بستهٔ ۱۰۲۴ بایتی یا دو بستهٔ ۵۱۲ بایتی یا چهار بستهٔ ۲۵۶ بایتی و به همین

ترتیب، ارسال کرد. اگر تعداد بایتهای باقیمانده ناچیز باشد، بستهٔ بعدی باید تا تیک ساعت بعدی منتظر بماند.

پیاده‌سازی الگوریتم سطل سوراخ (نسخهٔ اصلی آن) ساده است. سطل سوراخ متشکل از یک صف متناهی (محدود) است. هر گاه بسته‌ای دریافت شود و فضای کافی در صف وجود داشته باشد به آخر صف ملحق می‌شود و در غیر این صورت حذف می‌شود. در هر تیک ساعت یک بسته ارسال می‌شود (مگر آنکه صف خالی باشد).

«الگوریتم سطل سوراخ مبتنی بر شمارش بایت» نیز تقریباً به همین روش پیاده‌سازی می‌شود. در هر تیک ساعت شمارندهٔ بایت، به مقدار n تنظیم می‌شود. اگر بستهٔ سر صف تعداد بایت کمتری از مقدار فعلی شمارنده داشته باشد، ارسال شده و به تعداد بایتهای بسته از شمارنده کم می‌شود. مادامیکه مقدار شمارنده از بسته‌های موجود در سر صف بیشتر است می‌توان این بسته‌ها را ارسال کرد. وقتی مقدار شمارنده از اندازهٔ بستهٔ بعدی در سر صف کمتر شود، عمل ارسال تا تیک بعدی متوقف می‌شود تا مقدار شمارنده بایت مجدداً به مقدار n تنظیم شده و جریان بسته‌ها ادامه یابد.

به عنوان مثالی از الگوریتم سطل سوراخ، کامپیوتری را در نظر بگیرید که می‌تواند داده‌هایی با سرعت ۲۵ میلیون بایت در ثانیه (معادل 200 Mbps) تولید کند و شبکه نیز با همین سرعت کار می‌کند، ولیکن مسیریاب فقط می‌تواند چنین نرخ داده‌ای را در یک فاصله زمانی محدود بپذیرد (تا زمانی که بافرهایش پر شود). در زمانهای طولانی بهتر آن است که نرخ ارسال از ۲ میلیون بایت در ثانیه تجاوز نکند.[1] حال فرض کنید هر ۴۰ میلی‌ثانیه، داده‌ها در یک حالت انفجارگونه و بصورت توده‌های یک میلیون بایتی ارسال شوند. [یعنی هر ۴۰ میلی‌ثانیه یک تودهٔ یک میلیون بایتی با حداکثر سرعت یعنی ۲۵ میلیون بایت در ثانیه تولید و ارسال می‌شوند.] برای آنکه نرخ متوسط ارسال به 2MByte/Sec کاهش یابد، باید از «سطل سوراخ» با ρ=2MByte/sec و فضای بافر 1MByte بهره گرفته شود. این بدین معناست که توده‌های داده‌ای که انفجارگونه ارسال می‌شوند با طول حداکثر 1MByte قابل دریافت و مدیریت هستند و چیزی از دست نخواهد رفت. این سطل نیز در طول ۵۰۰ میلی‌ثانیه تخلیه می‌شود و اینکه داده‌ها با چه سرعتی وارد می‌شوند اهمیتی ندارد.

در شکل ۵-۳۳-الف ورود داده‌ها به سطل سوراخ در خلال ۴۰ میلی‌ثانیه و با نرخ 25 MByte/Sec صورت گرفته است [معادل ۱ مگابایت داده و متناسب با ظرفیت سطل]. در شکل ۵-۳۳-ب می‌بینیم که این مقدار از داده در خلال ۵۰۰ میلی‌ثانیه و با نرخ 2MByte/sec تخلیه و ارسال شده است.

الگوریتم سطل نشانه‌دار (The Token Bucket Algorithm)

الگوریتم سطل سوراخ الگوی خروجی سختگیرانه و با نرخ میانگین و ثابتی را تحمیل می‌کند و آنکه ترافیک تا چه اندازه انفجاری است برایش مهم نیست.[2] در بسیاری از کاربردها وقتی توده‌ای انفجاری از داده‌ها می‌رسد بهتر آن است که اجازه بدهیم سرعت تا حدّی افزایش یابد؛ بنابراین به الگوریتمی احتیاج داریم که قابلیت انعطاف بیشتری داشته باشد و ترجیحاً هیچ داده‌ای از دست نرود. یکی از این الگوریتمها **«الگوریتم سطل نشانه‌دار»** (Token Bucket Algorithm) است. در این الگوریتم، سطل سوراخ (یا بعبارتی بافر موجود در کارت شبکه یا

---

۱. یعنی اگرچه ظرفیت ارسال شبکه 25 Mbyte/Sec است ولی مسیریاب بدلیل محدودیت سرعت، فقط از عهدهٔ پردازش 2Mbyte/Sec برمی‌آید و در غیر اینصورت با ازدحام مواجه می‌شود. اگر فرستنده در لحظاتی با نرخ بیشتری ارسال کرد در عوض باید برای لحظاتی متوقف شود. -م

۲. یادآوری می‌کنیم که انفجاری بودن ترافیک بدین معناست که فقط در لحظات کوتاهی حجم انبوهی اطلاعات تولید و ارسال می‌شود و در بخش زیادی از زمان ارسال داده‌ها ناچیز است. فاکتور انفجاری بودن ترافیک را «حاصل تقسیم ماکزیمم ترافیک بر میانگین ترافیک» در نظر بگیرید. -م

سیستم عامل)، نشانه‌ای (اصطلاحاً یک **توکن**[1]) را نگه می‌دارد که این توکن در هر ΔT ثانیه یکبار تولید می‌شود. در شکل ۵-۳۴-الف ، سطلی را مشاهده می‌کنید که در آن سه نشانه (توکن) وجود دارد و همچنین پنج بسته منتظر ارسال هستند. هر گاه بسته‌ای بخواهد ارسال شود باید ابتدا یک نشانه بدست آورده و آن را از بین ببرد. در شکل ۵-۳۴-ب می‌بینیم که سه بسته از مجموع پنج بسته جواز خروج گرفته و ارسال شده‌اند در حالی که دو بستهٔ باقیمانده در انتظار تولید دو نشانهٔ دیگر به سر می‌برند.

نوع و مکانیزم شکل‌دهی به ترافیک در «الگوریتم سطل نشانه‌دار» نسبت به «الگوریتم سطل سوراخ» متفاوت است. الگوریتم سطل سوراخ به ماشینهای بیکار اجازه نمی‌دهد که وقتی بیکار هستند سهمیهٔ ارسال خود را نگه داشته و از این سهمیه به یکباره جهت ارسال انفجارگونهٔ بسته‌ها استفاده کنند. در طرف مقابل، الگوریتم سطل نشانه‌دار اجازه می‌دهد که ماشین سهمیهٔ خود را تا حداکثر حجم سطل (n بسته) پس‌انداز کنند. این ویژگی بدین معناست که توده‌ای انفجاری از بسته‌ها را (تا حداکثر n بسته) می‌توان یکجا فرستاد. [البته به شرط پس‌انداز تعداد n نشانه]. این ویژگی اجازه می‌دهد که خروجی بتواند تا حدودی حالت انفجارگونه داشته باشد و هنگامی که ورود بسته‌ها انفجاری است، پاسخ سریعتری داده شود.

تفاوت دیگر این دو الگوریتم آن است که الگوریتم سطل نشانه‌دار در هنگامی که سطل پر شود نشانه‌ها را دور می‌اندازد (یا به عبارتی ظرفیت ارسال را) در حالی که هرگز بسته‌ها را حذف نمی‌کند. در مقابل، الگوریتم سطل سوراخ به محض پر شدن سطل، بسته‌ها را حذف می‌نماید.

در اینجا یک تغییر کوچک ممکن است و آن اینکه هر نشانه (توکن) مجوز ارسال k بایت تلقی شود نه ارسال یک بسته. در این حالت یک بسته را فقط زمانی می‌توان ارسال کرد که به تعداد کافی نشانه (معادل با اندازهٔ بسته) پس‌انداز شده باشد. نشانه‌هایی که سهمیهٔ ارسال آنها (برحسب بایت) کوچکتر از اندازهٔ بستهٔ فعلی است برای استفادهٔ بعدی نگه داشته می‌شود.

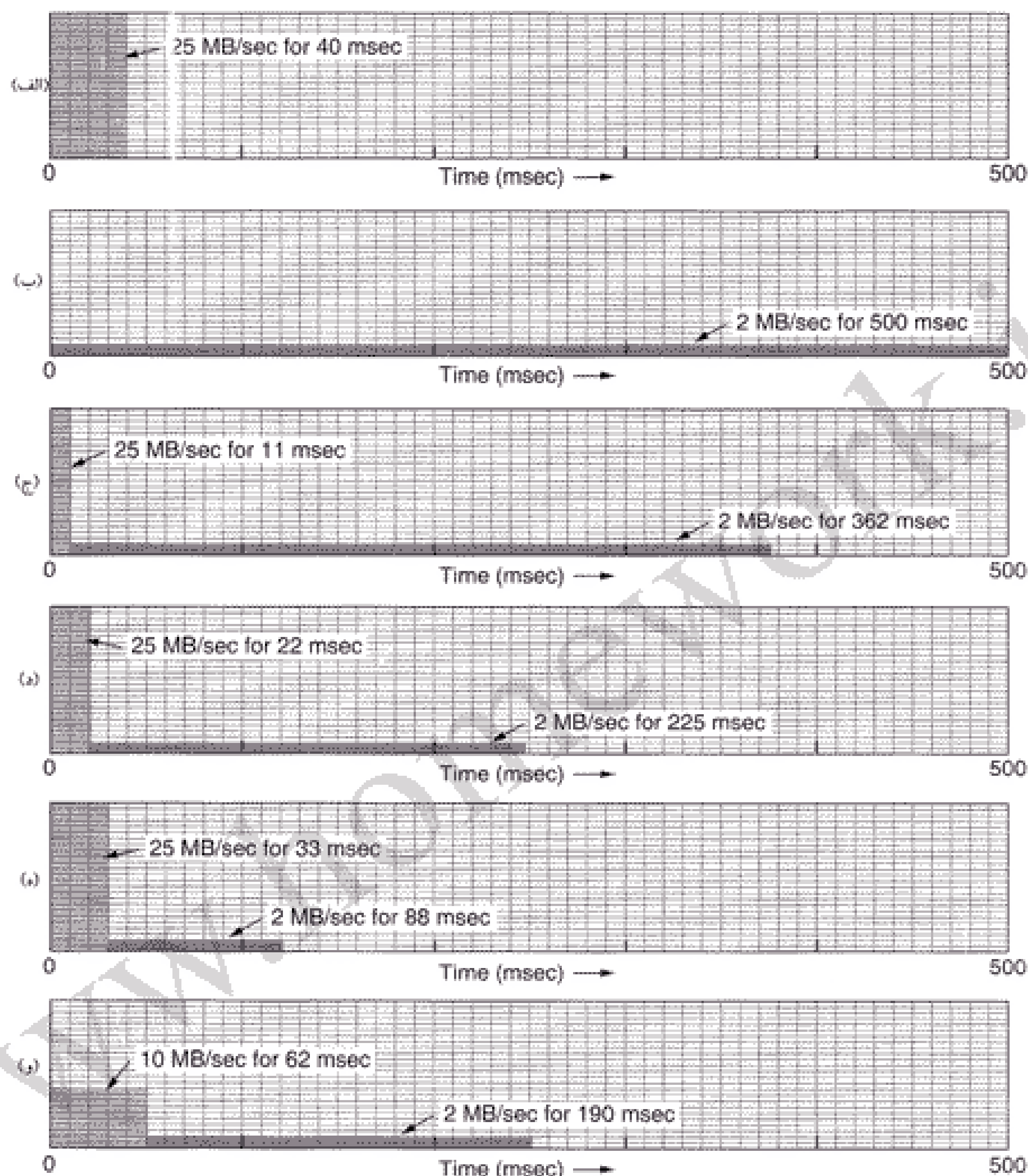
الگوریتم سطل سوراخ و الگوریتم سطل نشانه‌دار را می‌توان به همان نحوی که برای منظم کردن ترافیک خروجی ماشینهای میزبان بکار می‌رود برای متعادل کردن ترافیک بین مسیریابها نیز استفاده کرد. با این حال یک تفاوت بدیهی بین این دو کاربرد وجود دارد و آن هم اینکه الگوریتم سطل نشانه‌دار می‌تواند یک ماشین میزبان را وادار به توقف ارسال نماید، در حالی که وادار کردن یک مسیریاب به توقف (در حالی که بافرهای ورودی آن پر است) ، می‌تواند به از دست رفتن داده‌ها بینجامد.

پیاده‌سازی الگوریتم سطل نشانه‌دار، به سادگی تعریف یک متغیر است تا این متغیر تعداد نشانه‌ها (توکنها) را بشمارد. این شمارنده در هر ΔT ثانیه یک واحد افزایش داده شده و با ارسال یک بسته، کاهش می‌یابد. وقتی این شمارنده به صفر برسد هیچ بسته‌ای را نمی‌توان ارسال کرد. در گونهٔ دیگر این الگوریتم یعنی روش شمارش بایت، شمارنده هر ΔT ثانیه k واحد (k بایت) افزایش یافته و با ارسال بسته به اندازهٔ طول آن از شمارنده کسر می‌شود.

اصولاً کاری که الگوریتم سطل نشانه‌دار انجام می‌دهد آن است که اجازهٔ ارسال انفجارگونه بسته‌ها را صادر می‌کند ولیکن با سقف حداکثر و تنظیم‌شده‌ای که قابل کنترل و اداره باشد. برای مثال به شکل ۵-۳۳-ج دقت کنید. در اینجا یک سطل نشانه دار با ظرفیت ۲۵۰ کیلوبایت در اختیار داریم. نشانه‌ها (توکنها) متناسب با نرخ ۲ مگابایت بر ثانیه تولید می‌شوند. [یعنی نشانه‌ها با نرخی تولید می‌شوند که فقط اجازهٔ ارسال ۲ مگابایت اطلاعات در ثانیه را صادر می‌کنند. -م] فرض کنید هنگامی که یک تودهٔ انفجارگونهٔ ۱ مگابایتی تولید می‌شود، سطل پر است. سطل ، قادر است بمدّت حدود یازده میلی‌ثانیه و با سرعت 25MByte/sec ، داده‌های خود را خالی کند. از آن به بعد

---

۱. نشانه یا «توکن» را در اینجا جواز یا سهمیهٔ ارسال تعداد مشخصی بایت در نظر بگیرید. -م

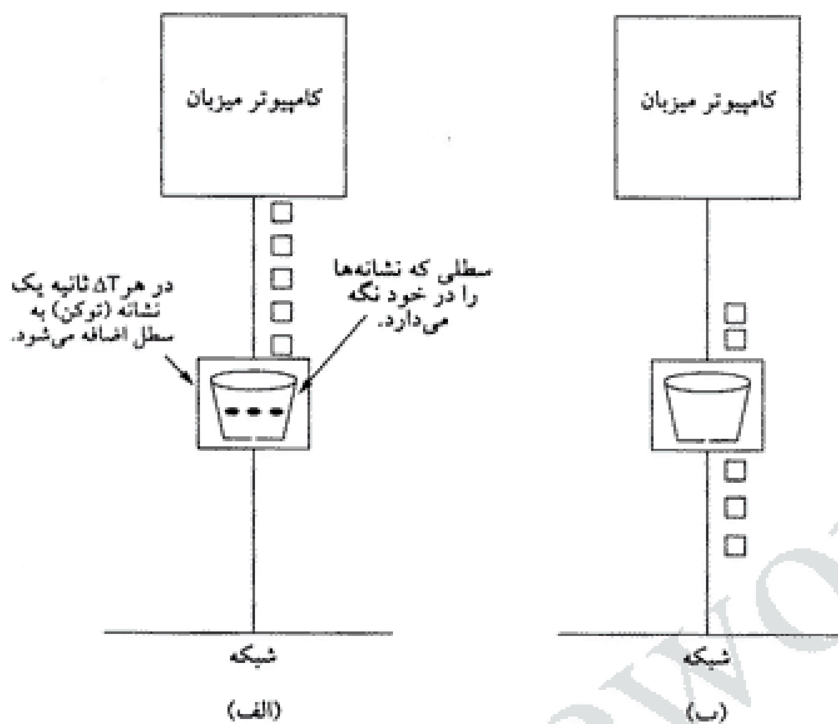


شکل ۵-۳۳. (الف) ورودی به یک سطل سوراخ‌دار. (ب) خروجی از سطل سوراخ‌دار. خروجی از یک سطل نشانه‌دار با ظرفیت: (ج) 250 KByte (د) 500 KByte (ه) 750 KByte. (و) خروجی یک سطل نشانه‌دار با ظرفیت 500KByte به یک سطل سوراخ‌دار با نرخ خروجی 10-Mbps اعمال شده است.

مجبور است تا وقتی که تودهٔ انفجاری داده ارسال نشده سرعت خروجی را بر روی 2MB/sec ثابت نگه دارد.[۱]

محاسبهٔ طول زمانی که می‌توان بصورت انفجاری و با نرخ حداکثر، تودهٔ تولید شده را ارسال کرد، اندکی پیچیده است. یعنی سقف حداکثر زمان ارسال انفجارگونه، معادل تقسیم «سهمیه موجود در سطل» بر «نرخ ارسال»

۱. یادآوری می‌کنیم که در الگوریتم سطل نشانه‌دار اگر بسته‌ها بیش از ظرفیت (سهمیهٔ) موجود در سطل تولید شوند، حذف نخواهند شد.



شکل ۵-۳۴. الگوریتم سطل نشانه‌دار. (الف) قبل از ورود بسته (ب) بعد از خروج بسته.

نیست چراکه در خلال ارسال این توده از بسته‌ها، نشانه‌های جدیدی تولید می‌شوند. اگر طول زمان ارسال انفجاری بسته‌ها را S بنامیم ، ظرفیت موجود در سطل نشانه‌دار C بایت ، نرخ تولید نشانه ρ بایت بر ثانیه و حداکثر نرخ خروجی M bytes/sec باشد، حداکثر نرخ ارسال انفجارگونهٔ خروجی $C+\rho.S$ بایت است.[1] همچنین می‌دانیم که اگر طول زمان ارسال، S ثانیه و حداکثر نرخ ارسال M بایت بر ثانیه باشد، تعداد بایتی که در این زمان ارسال می‌شود M.S بایت خواهد بود. بنابراین داریم:

$$C + \rho.S = M.S$$

با حل این معادله بدست می‌آوریم:

$$S = \frac{C}{M - \rho}$$

در مثال فوق با پارامترهای C=250 KByte ، ρ=2 MByte/sec و M=25 MByte/sec حداکثر زمان ارسال انفجارگونهٔ بسته‌ها یازده میلی‌ثانیه بدست خواهد آمد. شکلهای ۵-۳۳-د و ۵-۳۳-ه سطل نشانه‌دار را برای ظرفیت ۵۰۰کیلوبایت و ۷۵۰کیلوبایت نشان می‌دهد.

مشکل بالقوهٔ الگوریتم سطل نشانه‌دار آن است که کماکان اجازه می‌دهد توده‌ای از بسته‌ها به صورت انفجارگونه تولید و ارسال شود (حتی اگر طول زمان تولید به دقت و براساس انتخاب صحیح ρ و M تنظیم شده باشد). اغلب مطلوب آن است که نرخ حداکثر تولید بسته‌ها کاهش یابد ولی بدون آنکه بخواهیم به مقدار کم و ثابت

---

۱. یادآوری می‌کنیم که اگر سهمیهٔ موجود در سطل C بایت باشد، می‌توان C بایت از داده‌ها را با نرخ حداکثر و بصورت انفجاری ارسال کرد و پس از آن باید نرخ ارسال را متناسب با نرخ تولید نشانه کاهش داد. در خلال ارسال انفجاری این C بایت که بمدت S ثانیه طول می‌کشد تعدادی نشانهٔ جدید (سهمیهٔ جدید) تولید می‌شود و اجازه می‌دهد حجم ارسال انفجاری اندکی افزایش یابد. این حجم معادل $C+\rho.S$ بایت است: C بایت سهمیهٔ قبلی و $\rho.S$ بایت سهمیهٔ جدید تولید شده در زمان ارسال. -م.

الگوریتم سطل سوراخ بسنده نماییم.

یک روش برای متعادل کردن ترافیک آن است که بعد از «سطل نشانه‌دار» یک «سطل سوراخ» قرار داده شود. نرخ خروجی سطل سوراخ باید بیشتر از نرخ خروجی سطل نشانه‌دار (یعنی مقدار ρ) انتخاب شود ولی این مقدار باید از حداکثر نرخ مجاز شبکه کمتر باشد. شکل ۵-۳۳-و خروجی یک سطل نشانه‌دار با ظرفیت ۵۰۰کیلوبایت که پس از آن یک سطل سوراخ 10MByte/sec قرار داده شده را نشان می‌دهد.

نظارت بر این روشها ممکن است اندکی پیچیده باشد. اصولاً شبکه باید این الگوریتم را شبیه‌سازی کند و مطمئن گردد که بسته یا بایتهای بیشتر از حد مجاز و مشخص ارسال نمی‌شود. در هر حال این ابزارها روشهایی برای شکل‌دهی به ترافیک شبکه به گونه‌ای قابلیت مدیریت است تا بتوان نیازهای QoS (کیفیت خدمات) را برآورده نمود.

رزرو منابع (Resource Reservation)

توانایی شکل‌دهی و تنظیم ترافیک ارسالی ، تمهید خوبی برای تضمین «کیفیت خدمات» (QoS) محسوب می‌شود ولیکن استفاده از این روشها زمانی کارآمد خواهد بود که تمام بسته‌ها از مسیر یکسانی عبور کنند. پراکندگی تصادفی بسته‌ها بر روی مسیرهای متفاوت، تضمین هر چیزی را بسیار دشوار می‌کند. بنابراین برای تامین کیفیت خدمات باید بین مبداء و مقصد چیزی شبیه به یک مدار مجازی ایجاد و تنظیم شود و تمام بسته‌های یک «جریان» از این مسیر حرکت کنند.

هر گاه برای جریان داده‌ها، مسیر ویژه داشته باشیم می‌توان منابع لازم را در طول این مسیر، رزرو کرده و موجود بودن ظرفیت مورد نیاز را تضمین کرد. سه نوع متفاوت از منابع را می‌توان از قبل رزرو کرد:

۱. پهنای باند
۲. فضای بافر
۳. سیکلهای CPU [ظرفیت پردازش مورد نیاز]

رزرو پهنای باند آشکارترین مورد است: اگر یک جریان به 1Mbps پهنای باند نیاز داشته باشد و ظرفیت خط خروجی 2Mbps باشد تلاش برای هدایت سه جریان همزمان بر روی این خط عملی نیست. لذا رزرو پهنای باند به معنای آن نیست که می‌توان یک خط خروجی را بیش از ظرفیت آن رزرو کرد.

دومین منبع که اغلب با محدودیت مواجه است، فضای بافر می‌باشد. وقتی یک بسته دریافت می‌شود معمولاً توسط سخت‌افزار در حافظهٔ کارت شبکه ذخیره می‌گردد. بعداً نرم‌افزار مسیریاب باید آن را به یک بافر در فضای RAM اصلی منتقل کرده و پس از پردازش و تعیین مسیر، آنرا در صف مربوط به خط خروجی منتخب، وارد نماید. اگر هیچ بافری موجود نباشد، بستهٔ دریافتی حذف خواهد شد چرا که فضایی برای ذخیرهٔ آن موجود نیست. برای تضمین کیفیت خوب خدمات، باید برای هر «جریان» مشخص مقداری بافر کنار گذاشت تا لازم نباشد بسته‌های آن جریان برای بدست آوردن فضای بافر با دیگر بسته‌ها رقابت کنند. در این صورت به محض نیاز به بافر، فضای لازم در اختیار خواهد بود (البته تا حدّ معقول و مشخص).

در آخر، «سیکلهای CPU» نیز منبع کمیابی است: پردازش یک بسته به اندازهٔ معینی وقت مسیریاب را می‌گیرد لذا هر مسیریاب قادر است در یک ثانیه، تعداد مشخص و محدودی از بسته‌ها را پردازش کند. برای آنکه بتوان مطمئن شد که بسته‌ها در روالی منظم و به موقع پردازش می‌شوند باید مراقب بود که بار CPU بیش از حدّ زیاد نشود.

در نگاه اول شاید به نظر برسد که اگر پردازش هر بسته مثلاً یک میکروثانیه طول بکشد، پس مسیریاب

می‌تواند یک میلیون بسته را در هر ثانیه پردازش و هدایت نماید. چنین نتیجه‌گیری غلط است زیرا به دلیل تغییرات آماری بار [یعنی ثابت نبودن میزان بار]، پردازنده در دوره‌هایی از زمان بیکار می‌ماند. اگر CPU موظف باشد که در هر سیکل، کار پردازش را شروع و تکمیل کند، حتی از دست رفتن چند سیکل (ناشی از بیکاریهای مقطعی) انباشته شده و قابل جبران نخواهد بود.

با این حال حتی وقتی میزان بار کمتر از ظرفیت تئوریک مسیریاب باشد باز هم ممکن است صفّ تشکیل و تأخیر ایجاد شود. وضعیتی را در نظر بگیرید که در آن بسته‌ها بطور کاملاً تصادفی و با نرخ متوسط λ بسته بر ثانیه وارد می‌شوند. زمان مورد نیاز برای پردازش هر بسته نیز تصادفی است و بطور متوسط μ بسته در هر ثانیه پردازش می‌شود. با این فرض که تابع توزیع دریافت بسته‌ها و همچنین پردازش آنها «پوآسون» باشد می‌توان به کمک «نظریهٔ صف» (Queuing Theory) ثابت کرد که متوسط تأخیری که هر بسته با آن مواجه خواهد شد (یعنی T) معادل است با:

$$T = \frac{1}{\mu} \times \frac{1}{1-\lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1-\rho}$$

که در آن ρ معادل است با λ/μ و از آن به عنوان «میزان بهره‌وری CPU» (CPU Utilization) یاد می‌شود. فاکتور $1/\mu$ ، زمان لازم برای پردازش یک بسته است. (هنگامی که رقابت و صف وجود ندارد). فاکتور $1/(1-\rho)$ میزان کاهش سرعت پردازش، در اثر تشکیل صفّ (رقابت) تلقی می‌شود. به عنوان مثال اگر λ= 950000 Packet/sec و μ = 1,000,000 Packet/sec باشد میزان بهره‌وری CPU یعنی ρ معادل با ۰/۹۵ است و متوسط تأخیر بسته‌ها بجای ۱ میکروثانیه، ۲۰ میکروثانیه خواهد بود. این زمان در برگیرندهٔ زمان انتظار در صف و زمان سرویس دهی [زمان پردازش] بسته است. اگر مثلاً سی مسیریاب در مسیر جریان بسته‌ها قرار گرفته باشد زمان تأخیر انتظار صفّ در کل مسیر بتنهایی معادل با ۶۰۰ میکروثانیه است.

کنترل پذیرش (Admission Control)

حال در مرحله‌ای هستیم که ترافیک ورودی از یک «جریان» (Flow) خاصّ به خوبی شکل و نظم داده شده و بسته‌ها از یک مسیر واحد حرکت می‌کنند و پیشاپیش ظرفیت مورد نیاز در طول مسیر، پیش‌بینی و رزرو شده است. با چنین فرضی، هر گاه جریانی از بسته‌ها به یک مسیریاب تسلیم شود براساس ظرفیت موجود خود و سطح تعهداتی که در خصوص دیگر جریانها پذیرفته، باید در خصوص قبول یا ردّ آن تصمیم بگیرد.

تصمیم‌گیری در خصوص پذیرش یا ردّ یک «جریان»، یک مقایسه ساده بین میزان پهنای‌باند، بافر و سیکلهای CPU موردنیاز و ظرفیت باقیمانده در مسیریاب نیست. این تصمیم‌گیری اندکی پیچیده‌تر از این مقایسهٔ ساده است. اگرچه برخی از برنامه‌های کاربردی ممکن است میزان نیاز خود به پهنای باند را بدانند ولیکن آگاهی کمی از بافر یا حجم پردازش مورد نیاز بسته‌ها در مسیریابهای واقع بر روی مسیر دارند لذا حداقل می‌توان گفت که به روشی متفاوت برای توصیف و ارزیابی جریان نیاز است. در ثانی، برخی از کاربردها در مقابل برآورده نشدن انتظاراتشان ، تحمل بیشتری از خود نشان می‌دهند. در آخر آنکه ممکن است برخی از کاربردها بخواهند که در خصوص «پارامترهای جریان» چانه‌زنی و مذاکره کنند در حالی که امکان دارد برخی دیگر از کاربردها چنین خصوصیتی نداشته باشند. به عنوان مثال یک برنامهٔ نمایش‌دهندهٔ فیلم که معمولاً ۳۰ فریم تصویر در ثانیه را نمایش می‌دهد ممکن است بخواهد در صورت عدم وجود پهنای باند کافی برای این حجم از داده، تعداد فریمهای تصویر را تا 25 Frame/sec کاهش بدهد. به همین ترتیب ممکن است تعداد نقاط تصویر در هر فریم (Pixel Per Frame)، پهنای باند صدا و دیگر ویژگیها نیز قابل تغییر باشد.

چونکه برای رسیدن به توافق نهایی در خصوص تامین نیازهای یک «جریان»، باید مولفه‌های متعددی در

مذاکرات شرکت داشته باشند (اعمّ از فرستنده، گیرنده و تمام مسیریابهای واقع بر روی مسیر)، لذا هر «جریان» باید برحسب پارامترهای مشخصی بدقت توصیف شود تا بتوان بر روی این پارامترها مذاکره و توافق کرد. مجموعهٔ چنین پارامترهایی اصطلاحاً **«مشخصات توصیفی جریان»** (Flow Specification) نامیده می‌شود. بدین ترتیب یک فرستنده (مثل سرویس‌دهندهٔ ویدیو) مشخصات توصیفی جریان را به صورت پارامترهای پیشنهادی و مورد نظر خود تعریف می‌نماید. این پارامترهای پیشنهادی در طول مسیر منتشر می‌شود و هر مسیریاب واقع بر مسیر آنها را بررسی کرده و در صورت نیاز در آنها تغییراتی ایجاد می‌کند. این تغییرات فقط کاهشی است نه افزایشی (یعنی مثلاً نرخ مورد نظر ارسال داده‌ها را کاهش می‌دهد نه افزایش). وقتی این پارامترها به طرف مقابل برسد ، به اجرا گذاشته می‌شوند.

در مثال شکل ۵-۳۵، نمونه‌ای از پارامترهای توصیف جریان، مبتنی بر RFC 2210 و RFC 2211 نشان داده شده است. در اینجا پنج پارامتر وجود دارد که اولین آنها پارامتر Token Bucket Rate (نرخ تولید نشانه در الگوریتم سطل نشانه‌دار) است و تعداد بایتهایی را مشخص می‌کند که در هر ثانیه به سطل وارد می‌شوند. در حقیقت این پارامتر حداکثر نرخ ارسال مُجاز و قابل قبولی است که فرستنده می‌تواند ارسال نماید و میانگین ارسال در یک زمان طولانی تلقی می‌شود.

پارامترهای توصیف جریان	واحد
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

شکل ۵-۳۵. مثالی از توصیف جریان.

پارامتر دوم، حجم سطل را برحسب بایت مشخص می‌کند. به عنوان مثال اگر پارامتر Token Bucket Rate یک مگابیت در ثانیه و حجم سطل 500KByte باشد حتی در صورت توقف ارسال، این سطل می‌تواند تا چهار ثانیه پر شود و پس از آن سرریز خواهد شد. [سطلی با گنجایش ۵۰۰کیلوبایت معادل ۴ مگابیت، می‌تواند جریانی از داده‌ها با نرخ 1Mbps را بمدت ۴ ثانیه بافر کند. -م]

پارامتر سوم یا Peak Data Rate (حداکثر نرخ ارسال) بیانگر حداکثر نرخ قابل تحمل حتی در بازه‌های زمانی کوتاه است. فرستنده هیچگاه نباید از این نرخ تجاوز نماید.

دو پارامتر آخر مقدار حداقل و حداکثر طول بسته‌ها را مشخص می‌نماید. (این طول شامل سرآیند بستهٔ لایهٔ شبکه و لایهٔ انتقال است.) حداقل طول بسته‌ها نیز اهمیت دارد چراکه پردازش هر بسته (حتی اگر کوچک باشد) مدت زمان ثابتی طول می‌کشد. یک مسیریاب ممکن است آمادگی پذیرش و هدایت ۱۰,۰۰۰ بستهٔ یک کیلوبایتی در هر ثانیه را داشته باشد ولیکن آمادگی پذیرش و هدایت ۱۰۰,۰۰۰ بستهٔ ۵۰بایتی در ثانیه را نداشته باشد، هر چند که مورد دوم از لحاظ حجم داده کمتر از مورد اول است. طول حداکثر هر بسته نیز به دلیل محدودیتهای درونی شبکه اهمیت دارد و نباید از این مقدار حداکثر تجاوز شود. به عنوان مثال اگر بخشی از مسیر از درون شبکهٔ اترنت بگذرد، حداکثر طول بسته نباید بیش از ۱۵۰۰ بایت باشد [به دلیل معماری سخت‌افزار اترنت].

یک سؤال جالب آن است که یک مسیریاب با استفاده از مشخصات توصیفی جریان، چگونه می‌تواند منابع مورد نیاز را رزرو نماید. نگاشت «مشخصات توصیفی جریان» به میزان منابع مورد نیاز، در مقولهٔ پیاده‌سازی

می‌گنجد و استانداردسازی نشده است. فرض کنید که یک مسیریاب قادر به پردازش صد هزار بسته در ثانیه باشد. اگر «جریان» پیشنهادی با نرخ 1MB/sec و مقدار حداقل و حداکثر طول بسته‌ها ۵۱۲ بایت باشد، مسیریاب بدین نتیجه می‌رسد که از این جریان در هر ثانیه ۲۰۴۸ بسته دریافت خواهد کرد و بدین منظور ۲ درصد از CPU خود را برای پردازش این بسته‌ها در نظر می‌گیرد. ترجیحاً این مقدار باید بیشتر هم باشد تا از تأخیرات ناشی از ایجاد صفهای طولانی احتراز شود. اگر سیاستهای یک مسیریاب بر این اساس باشد که هیچگاه بیش از ۵۰ درصد از وقت CPU خود را اختصاص ندهد (یعنی تلویحاً تأخیرات را دو برابر فرض کرده است) و قبلاً ۴۹ درصد از این وقت رزرو شده باشد نمی‌تواند یک «جریان» با مشخصات فوق را بپذیرد [چرا که به دو درصد از CPU احتیاج دارد در حالی که فقط یک درصد باقیمانده است]. برای منابع دیگر [مثل حافظه و پهنای باند] نیز محاسبات و پیش‌بینی‌های مشابه فوق انجام می‌شود.

توصیف دقیقتر «جریان»، برای مسیریابها مفیدتر خواهد بود. اگر در توصیف یک «جریان» بیان شود که پارامتر Token Bucket Rate (نرخ تولید نشانه در الگوریتم سطل) معادل 5MB/sec است ولیکن طول بسته‌ها بین ۵۰ تا ۱۵۰۰ بایت متغیر اعلام شود در این توصیف نادقیق، نرخ ارسال بسته‌ها بین ۳۵۰۰ تا ۱۰۵,۰۰۰ بسته در ثانیه متغیر خواهد بود و ممکن است مسیریاب از عدد ۱۰۵۰۰۰ نگران شده و چنین جریانی را نپذیرد در حالی که اگر مقدار حداقل طول بسته ۱۰۰۰ بایت پیشنهاد شود احتمالاً این جریان پذیرفته خواهد شد.

مسیریابی نسبی

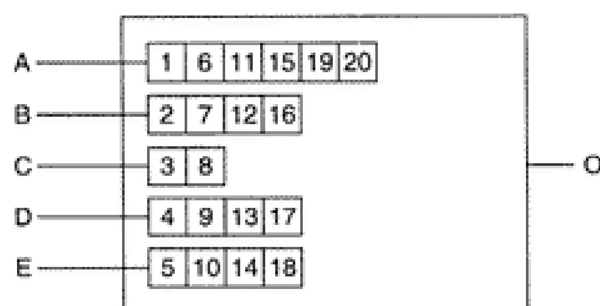
در اغلب الگوریتمهای مسیریابی سعی بر آن است که بهترین مسیر به هر مقصد پیدا شود و پس از آن تمام ترافیک به یک مقصد از مسیر بهینه ارسال خواهد شد. راهکار دیگر برای ارائه کیفیت بهتر خدمات آنست که ترافیک بسته‌های ارسالی برای یک مقصد، از چندین مسیر هدایت و ارسال شود. از آنجایی که مسیریابها دید جامعی از ترافیک سرتاسر شبکه ندارند لذا تنها راه ممکن برای توزیع ترافیک بر روی چندین مسیر، استفاده از اطلاعات موجود و محلی است. یک روش ساده آن است که ترافیک بطور مساوی یا به تناسب ظرفیت هر یک از خطوط خروجی مسیریاب، بر روی آنها توزیع شود. با این حال الگوریتمهای پیچیده‌تری در این خصوص وجود دارد. (Nelakuditi and Zhang, 2002)

زمان بندی بسته‌ها

هر گاه یک مسیریاب هدایت چندین «جریان» را بر عهده داشته باشد این خطر وجود دارد که یک «جریان» از حدود و ظرفیت مجاز خود تجاوز نماید و در نتیجه جریانهای دیگر را با کمبود منابع (starvation) مواجه سازد. اگر پردازش بسته‌ها به ترتیب ورودشان انجام گیرد باعث می‌شود که یک فرستندهٔ متجاوز بتواند بیشتر ظرفیت مسیریابهایی را که بر روی خط سیر بسته‌های او هستند اشغال کرده و کیفیت خدمات دیگران کاهش یابد. برای خنثی کردن چنین تلاشی، الگوریتمهایی جهت زمان‌بندی بسته‌ها پیشنهاد شده است. (Bhatti and Crowcroft, 2000)

یکی از اولین روشها، الگوریتم «صف‌بندی بی‌طرفانه» (Fair Queuing) است. (Nagle, 1987) جوهرهٔ این الگوریتم آنست که مسیریابها باید برای هر خط خروجی و به ازای هر «جریان» که از آن خط خروجی می‌گذرد، صفهای جداگانه‌ای تشکیل بدهند. هر گاه خطی بیکار شود، مسیریاب صفّها را به ترتیب پویش کرده و از سر هر صفّ یکی را بر می‌دارد. بدین ترتیب، در شرایطی که n ماشین میزبان برای یک خط خروجی رقابت می‌کنند، از هر n بستهٔ ارسالی بر روی خط یک بسته به هر ماشین میزبان تعلق می‌گیرد. افزایش نرخ ارسال بسته‌ها، در نسبت سهم هر ماشین تغییری ایجاد نخواهد کرد.

البته در همین ابتدا، الگوریتم فوق دارای یک مشکل است: آن ماشینهای میزبان که طول بسته‌هایشان بزرگ است نسبت به ماشینهایی که بسته‌های کوچک تولید می‌کنند، سهم بیشتری از پهنای باند را بخود اختصاص خواهند داد. پژوهشگری به نام Demer و همکاران او (۱۹۹۰) پیشنهادی جهت بهبود این الگوریتم ارائه دادند. پیشنهاد آن بود که بجای ارسال یک بسته از هر صف به صورت نوبت چرخشی (Round Robin)، سهم ارسال هر صف بر حسب بایت باشد. در نتیجه صفها بطور متوالی و بایت به بایت پویش شده و بسته‌های هر صف برحسب زمان خاتمهٔ ارسالشان مرتب شده و بهمان ترتیب ارسال می‌شوند. یعنی به جای آنکه از هر صف فقط یک بسته انتخاب شود تعدادی بایت و متناسب با سهم زمانی هر صف ارسال می‌شود. این الگوریتم در شکل ۵-۳۶ به تصویر کشیده شده است.



(الف)

زمان خاتمه ارسال	بسته
8	C
16	B
17	D
18	E
20	A

(ب)

شکل ۵-۳۶. (الف) مسیریابی که در آن پنج بسته برای خروج از خطّ O به صف شده‌اند. (ب) زمان خاتمهٔ ارسال این پنج بسته.

در شکل ۵-۳۶-الف بسته‌هایی به طول ۲ تا ۶ بایت می‌بینیم. در هر تیک ساعت (مجازی) اولین بایت از بستهٔ دریافتی از خط A ارسال می‌شود. در تیک بعدی اولین بایت از بستهٔ دریافتی از خط B ارسال می‌گردد و کار به همین ترتیب ادامه می‌یابد. اولین بسته‌ای که ارسال آن پس از هشت تیک ساعت خاتمه خواهد یافت بستهٔ C است.[1] در شکل ۵-۳۶-ب فهرست مرتب شدهٔ بسته‌ها به ترتیب ارسال مشخص شده است. هر گاه بستهٔ جدیدی دریافت نشود بسته‌ها به ترتیب فوق‌الذکر (از C تا A) ارسال خواهد شد.

یک اشکال این الگوریتم آن است که به تمام ماشینهای میزبان، اولویت یکسانی می‌دهد. در بسیاری از محیطها مطلوبتر آن است که به سرویس‌دهنده‌های ویدیو (Video Server) اولویت بیشتری نسبت به یک سرویس‌دهندهٔ معمولی فایل داده شود و در هر تیک ساعت، سهم آن دو یا چند بایت باشد. این الگوریتم اصلاح شده به نام «الگوریتم صف‌بندی بی‌طرفانهٔ وزن‌دار» (Weighted Fair Queuing) مشهور است و کاربرد گسترده‌ای دارد. گاهی اوقات وزن هر صف معادل با تعداد «جریان» (Flow) منشعب از یک ماشین در نظر گرفته می‌شود و بدین ترتیب هر پروسهٔ مولد جریان، پهنای باند یکسانی دریافت می‌دارد.[2] روش پیاده‌سازی مؤثر این الگوریتم در مرجع (Shreedhar & Varghese, 1995) تشریح شده است. امروزه در عمل، هدایت بسته‌ها توسط سخت‌افزار مسیریاب یا سوئیچ انجام می‌شود و الگوریتمهای فوق برروی سخت‌افزاری پیاده‌سازی شده‌اند. (Elhanany et al., 2001)

۱. دقت کنید که شماره‌هایی که درون مربعهای هر بسته از شکل ۵-۳۶ نوشته شده شمارهٔ ترتیب ارسال تلقی می‌شود و هر مربع شماره‌دار صرفاً معادل یک بایت است. -م

۲. به عبارت دیگر به جای آنکه به ماشینهای میزبان پهنای باند مساوی داده شود به پروسه‌های هر ماشین پهنای باند یکسان داده می‌شود. -م

۵-۴-۳ خدمات مجتمع (Integrated Services)

در خلال سالهای ۱۹۹۵ تا ۱۹۹۷، تلاش IETF بر آن بود که برای انتقال داده‌های مالتی‌مدیا (Multimedia Streaming) معماری مناسبی ابداع کند. نتیجهٔ کار چندین RFC به شماره‌های ۲۲۰۵ تا ۲۲۱۰ بود. این پروژه با نام کلّی «**الگوریتمهای مبتنی بر جریان**» (Flow-based algorithms) یا «**خدمات مجتمع**» (Integrated Services) شناخته می‌شود و کاربردهای چندپخشی (Multicast) و تک‌پخشی (Unicast) را در بر می‌گیرد. به عنوان یک نمونه از کاربردهای تک‌پخشی، کاربری را در نظر بگیرید که قطعه‌ای ویدیو را از یک سایت تماشا می‌کند. به عنوان مثالی از کاربردهای چندپخشی، ایستگاههای پخش تلویزیون دیجیتال را در نظر بگیرید که برنامه‌های خود را در قالب جریانی از بسته‌های IP به گیرندگان بی‌شمار و پراکندهٔ خود ارسال می‌دارند. در ادامه بر روی کاربردهای چندپخشی متمرکز خواهیم شد چراکه کاربردهای تک‌پخشی حالت خاص چندپخشی هستند.

در اغلب کاربردهای چندپخشی، گروههای مختلف می‌توانند به صورت پویا عضویت خود را تغییر بدهند: مثلاً افرادی را مجسم کنید که در یک کنفرانس ویدیویی وارد می‌شوند و پس از مدتی حوصلهٔ آنها سر می‌رود و به یک کانال پخش آواز می‌پیوندند! در چنین شرایطی روش رزرو پهنای باند به خوبی کار نخواهد کرد چرا که هر فرستنده باید دائماً ورود و خروج اعضای خود را پیگیری نماید. برای سیستمی که مثلاً برای پخش تلویزیونی طراحی شده و میلیونها مشترک دارد این روشها هرگز کار نخواهد کرد!

RSVP[1] : پروتکل رزرو منابع

اصلیترین پروتکل پیشنهاد شده توسط IETF برای ارائه خدمات مجتمع، RSVP نامیده می‌شود. این پروتکل در RFC 2205 تشریح شده است و برای رزرو کردن پهنای باند بکار می‌آید. پروتکلهای دیگری که در RFC 2206 تا RFC 2210 تشریح شده‌اند چگونگی ارسال داده‌ها را توصیف می‌کنند. RSVP اجازه می‌دهد که چندین فرستنده بتواند برای چندین گروه از گیرندگان خود داده بفرستند و همچنین امکان آن را فراهم کرده که گیرندگان بتوانند کانال مورد نظر خود را آزادانه عوض کنند. در عین حال پروتکل RSVP، استفاده از پهنای باند را بهینه‌سازی کرده و از بروز ازدحام جلوگیری می‌کند.

در ساده‌ترین حالت، این پروتکل از روش «مسیریابی چندپخشی مبتنی بر درخت پوشا»[2] که قبلاً تشریح شد، بهره می‌گیرد. به هر گروه یک آدرس یکتا انتساب داده می‌شود و برای ارسال یک بسته به گروه خاصّ، آدرس آن گروه در بسته قرار می‌گیرد. سپس توسط الگوریتم استاندارد مسیریابی چندپخشی، یک درخت پوشا که تمام اعضای آن گروه را در بر می‌گیرد، ایجاد می‌گردد. الگوریتم مسیریابی چندپخشی جزو استاندارد RSVP محسوب نمی‌شود و تنها تفاوت آن با الگوریتم معمولی مسیریابی چندپخشی آنست که بطور متناوب، مقداری اطلاعات اضافی برای هر گروه ارسال می‌شود تا مسیریابهای واقع بر روی درخت، آنها را در ساختمان دادهٔ خاصّی در حافظهٔ خود ذخیره نمایند.

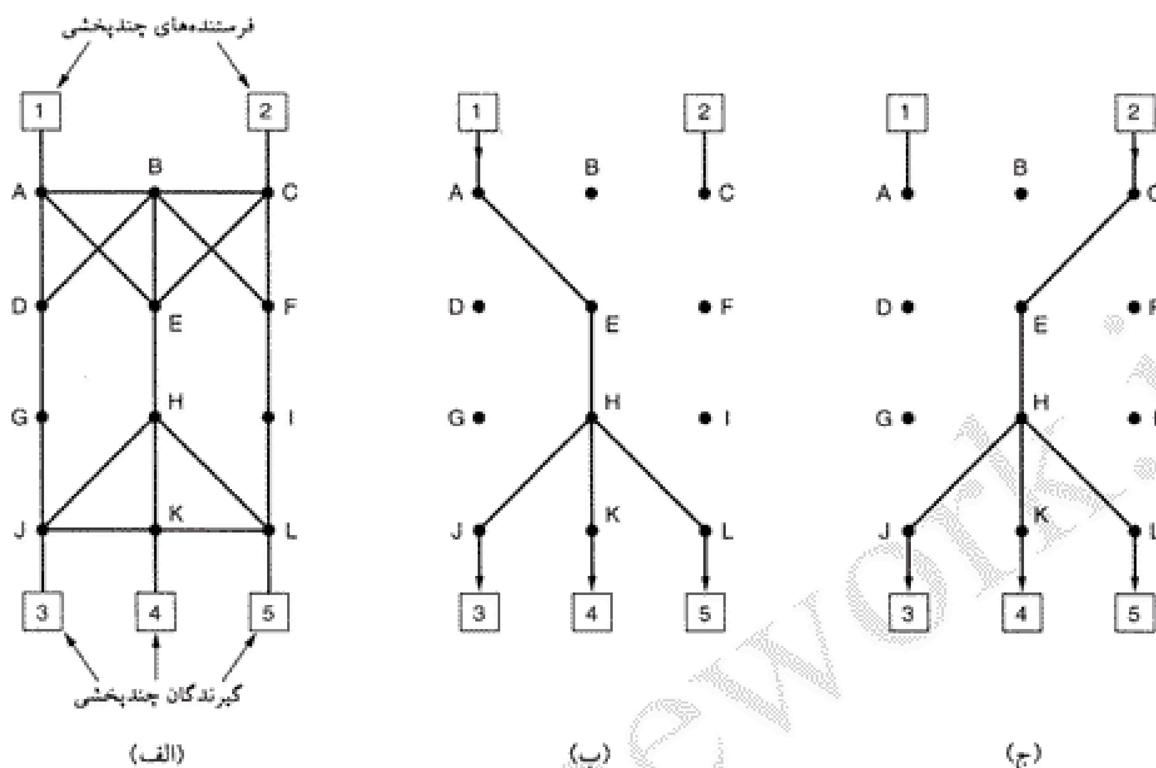
به عنوان مثال شبکهٔ شکل ۵-۳۷-الف را در نظر بگیرید. ماشینهای میزبان ۱ و ۲ فرستنده‌های چندپخشی و ماشینهای ۳ و ۴ و ۵ گیرندگان چندپخشی هستند. در این مثال فرستنده‌ها و گیرنده‌ها کاملاً از هم جدا (Disjoint) هستند ولی در حالت کلی ممکن است مجموعهٔ ماشینهای فرستنده و گیرنده عضو مشترک هم داشته باشند. درختهای چندپخشی برای فرستندهٔ شماره ۱ و شماره ۲ در شکلهای ۵-۳۷-ب و ۵-۳۷-ج نشان داده شده‌اند.

برای دریافت بهتر و جلوگیری از ازدحام، هر یک از گیرندگان یک گروه می‌توانند پیامی برای رزرو پهنای باند

---

۱. Resource Reservation Protocol

۲. Spanning Tree Multicast Routing (به بخش ۵-۲-۸ مراجعه کنید.)



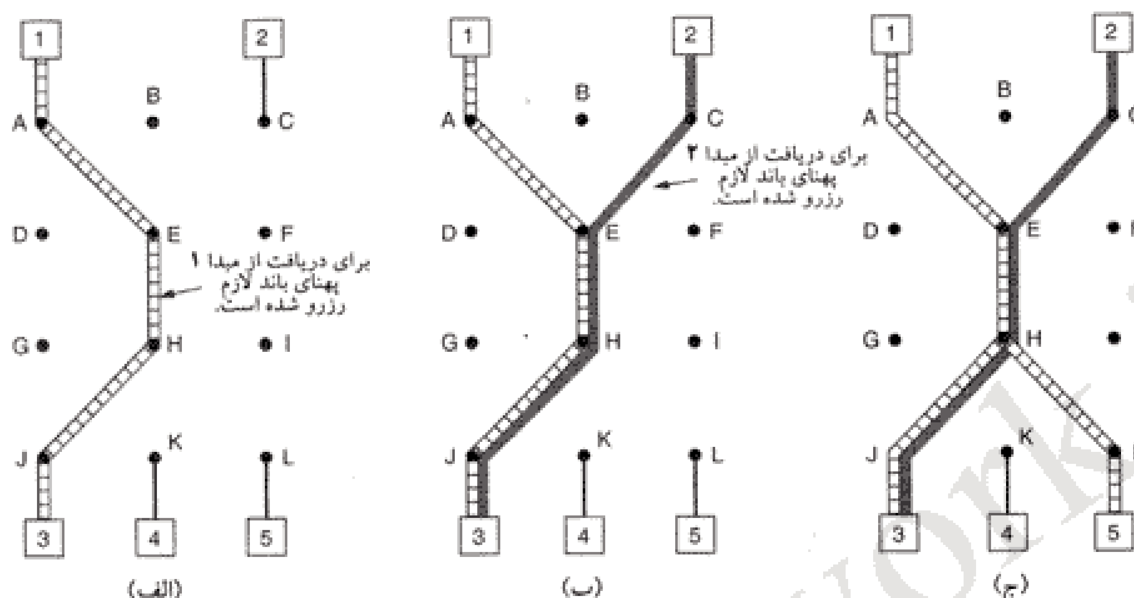
شکل ۵-۳۷. (الف) ساختار یک شبکه (ب) «درخت پوشای چندپخشی» برای ماشین میزبان ۱ (ج) «درخت پوشای چندپخشی» برای ماشین میزبان ۲.

ارسال کنند تا از طریق درخت به فرستنده برسد. این پیام به کمک «الگوریتم هدایت در مسیر معکوس»[1] که قبلاً تشریح شد، در درخت انتشار می‌یابد. در هر گام، مسیریاب به این پیام توجه کرده و پهنای باند لازم را رزرو می‌نماید. اگر پهنای باند کافی موجود نباشد، «پیغام شکست» برخواهد گشت. اگر این پیغام موفقیت‌آمیز به مبداء آن برگردد، پهنای باند لازم در کل مسیر رزرو شده است.

مثالی از چگونگی رزروسازی در شکل ۵-۳۸-الف نشان داده شده است. در اینجا ماشین میزبان ۳ تقاضای برقراری یک کانال با ماشین ۱ را داده است. به محض برقراری چنین کانالی، بسته‌ها می‌توانند بدون مسئلهٔ ازدحام از ۱ به ۳ جاری شوند. حال ببینیم اگر ماشین ۳ بعداً بخواهد کانالی دیگر با ماشین میزبان ۲ برقرار کند چه اتفاقی می‌افتد (زیرا مثلاً می‌خواسته دو کانال تلویزیونی را بطور همزمان تماشا کند.) مسیر دوم مطابق شکل ۵-۳۸-ب رزرو می‌شود. دقت کنید که از ماشین گیرندهٔ ۳ تا مسیریاب E به دو کانال مستقل نیاز است چراکه قرار است دو «استریم ویدیویی مستقل»[2] ارسال شود.

در شکل ۵-۳۸-ج ، ماشین میزبان ۵ تصمیم می‌گیرد برنامه‌ای را که توسط ماشین ۱ پخش می‌شود تماشا کند و او نیز رزرواسیون لازم را انجام می‌دهد. ابتدا پهنای باند لازم تا H رزرو می‌شود. از H به بعد، مسیریابها متوجه می‌شوند که از قبل کانالی با ماشین ۱ دارند و از داده‌های آن تغذیه می‌شوند، لذا نیازی به رزرو نیست. دقت کنید که ممکن است ماشینهای ۳ و ۵ به پهنای باند متفاوتی نیاز داشته باشند (چرا که مثلاً ماشین ۳ تصاویر تلویزیونی را سیاه و سفید نگاه می‌کند و نیازی به اطلاعات رنگی ندارد)، فلذا ظرفیت رزرو شده باید آنقدر زیاد باشد تا بتواند از گیرنده‌ای را که به بیشترین مقدار پهنای باند احتیاج دارد، برآورده سازد.

۱. Reverse Path Forwarding (به بخش ۵-۲-۷ مراجعه کنید.)
۲. Independent Stream



شکل ۵–۳۸. (الف) ماشین میزبان ۳ تقاضای کانالی به ماشین ۱ می‌کند. (ب) ماشین میزبان ۳ تقاضای کانالی دیگر به ماشین ۲ می‌کند. (ج) ماشین میزبان ۵ تقاضای کانالی به ماشین ۱ می‌کند.

در RSVP به هر گیرنده آزادی عمل داده شده تا بتواند با یکبار عملیات رزرو، کانالهایی با بیش از یک فرستنده، ایجاد نماید. همچنین می‌تواند مشخص کند که آیا انتخاب او برای دوره‌ای از زمان ثابت است یا آنکه حق تغییر مبداء ارسال (فرستنده) را برای خود محفوظ نگاه داشته است. مسیریاب می‌تواند به کمک این اطلاعات برآورد بهینه و مناسبی از پهنای باند داشته باشد.

دلیل اتخاذ این استراتژی در محیطهای کاملاً پویا و در حال تغییر آنست که پهنای باند رزرو شده از انتخاب مبداء (فرستنده) مستقل و مجزاست: به محض آنکه یک گیرنده که قبلاً پهنای باند لازم را برای دریافت از یک مبداء خاص رزرو کرده، بخواهد مبداء دریافت خود را تغییر بدهد بخشی از مسیر قبلی برای مبداء جدید نیز معتبر است و نیازی به تخصیص پهنای باند ندارد. در ضمن اگر ماشین ۲ در حال ارسال همزمان چندین جریان ویدیویی (Video Streams) باشد، ماشینی مثل ۳ می‌تواند بدون نیاز به هیچگونه رزرواسیون مجدد و تغییر، از بین این کانالهای ویدیویی یکی را انتخاب نماید زیرا مسیریابها به آنچه که گیرنده تماشا می‌کند دقت و اعتنایی نمی‌کنند.[۱]

۵-۴-۴ خدمات متمایز (Differentiated Services)

«الگوریتمهای مبتنی بر جریان»[۲] قابلیت عرضهٔ کیفیت خوب خدمات به یک یا چند جریان را دارند زیرا در طول مسیر هر منبعی را که نیاز است از قبل رزرو می‌کنند. ولی این روشها یک اشکال دارند: در این الگوریتمها نیاز است که برای هر جریان (Flow)، پیشاپیش تنظیمات لازم انجام شود در حالی که در مقیاس کلان یعنی وقتی که هزاران یا میلیونها «جریان» وجود دارد قابلیت اجرایی خود را از دست می‌دهند. از طرفی در هر مسیریاب «وضعیت» هر

۱. بعبارت روشنتر وقتی یک گیرنده مثل A کانالی با پهنای باند معین با فرستندهٔ B رزرو می‌کند و B بطور همزمان مثلاً ده استریم ویدیویی پخش می‌کند، گیرندهٔ A در انتخاب یکی از این ده استریم آزادی عمل دارد و نیازی نیست که به مسیریابها در این خصوص اطلاع داده شود چراکه آنها پهنای باند لازم را رزرو کرده‌اند و اصراری ندارند که بدانند فرستنده چه چیزی برای گیرنده می‌فرستد. ـم

۲. Flow-Based Algorithms

جریان بطور جداگانه نگهداری می‌شود و عملکرد این الگوریتم‌ها در مقابل خرابی یک مسیریاب آسیب‌پذیر خواهد بود. نهایتاً آنکه برای تنظیم و ایجاد «جریان» باید تبادل اطلاعات پیچیده‌ای بین مسیریابها انجام گیرد. در نتیجه RSVP یا الگوریتمهای مشابه آن، بسیار کم پیاده‌سازی عملی شده‌اند.

به همین دلایل، IETF راهکاری ساده‌تر برای تأمین کیفیت خدمات (QoS) ابداع کرد؛ روشی که بدون نیاز به هیچ تنظیمات قبلی یا تعیین کلّ مسیر، می‌تواند به صورت محلی و مجزا در هر مسیریاب پیاده‌سازی شود. این راهکار اصطلاحاً «**روش مبتنی بر کلاس**» (Class-Based) برای تضمین کیفیت خدمات نامیده می‌شود (در مقابل روشهای مبتنی بر جریان). IETF یک معماری مناسب به نام «**خدمات متمایز**» برای آن طراحی و استانداردسازی کرده است که در مستندات RFC به شماره‌های ۲۴۷۴ و ۲۴۷۵ و مستندات دیگر تشریح شده است. در ادامه به تشریح این روش می‌پردازیم.

«خدمات متمایز» (که به اختصار DS گفته می‌شود) می‌تواند توسط مجموعه‌ای از مسیریابها که در یک «حوزهٔ مدیریتی واحد» (Administrative Domain) قرار می‌گیرند (مثلاً یک ISP یا شرکت مخابرات)، عرضه شود. مدیریت مسئول شبکه، مجموعه‌ای از کلاسهای متفاوت خدمات و متناظر با آن، قواعد هدایت بسته‌ها (Forwarding Rules) را تعریف می‌کند.

اگر یک مشتری برای دریافت خدمات نوع DS تقاضای ورود به شبکه را بدهد، بسته‌های ارسالی او در ورود به حوزه، فیلد «نوع خدمات» (Type of Service) را با خود حمل می‌کنند تا به برخی از آنها خدمات بهتری (مثل **خدمات ویژه**[1]) ارائه شود. ممکن است لازم باشد ترافیک تعریف شده در هر کلاس از شکل خاصی پیروی نماید (مثلاً باید از الگوریتم سطل سوراخ با نرخ خروجی مشخص تبعیت کند). متصدی شبکه با گرایشات اقتصادی و تجاری ممکن است برای انتقال «بسته‌های ویژه» (Premium Packets) هزینهٔ اضافی بگیرد یا مثلاً به ازای بهای اشتراک ثابت و ماهانه، تعداد N بستهٔ ویژه از کاربر پذیرفته و هدایت شود. دقت کنید که این الگو نیاز به تنظیمات قبلی، رزروسازی منبع و نیازی به اتلاف وقت برای مذاکره بین طرفین نهایی در هر «جریان» ندارد. به همین دلیل پیاده‌سازی خدمات DS بسیار آسان است.

«خدمات مبتنی بر کلاس» در صنایع دیگر نیز وجود دارد. به عنوان مثال، شرکتهای تحویل محموله‌های پستی (Package Delivery) نیز سه نوع خدمات عرضه می‌کنند: شبانه روزی، دو روزه یا سه روزه. یا مثلاً خطوط هواپیمایی «خدمات کلاس برتر» (First Class)، «کلاس تجاری» (Business Class) و «کلاس معمولی» عرضه می‌کنند. در قطارهای دوربیما نیز خدمات در کلاسهای متفاوتی عرضه می‌شود و حتی قطارهای زیرزمینی (مترو) نیز در دو کلاس مختلف خدمات ارائه می‌کنند. برای بسته‌های حاوی اطلاعات، کلاسهای متفاوت خدمات برحسب میزان «تأخیر»، «لرزش» (Jitter)، احتمال حذف بسته در صورت بروز ازدحام و امکاناتی نظیر همینها تعیین می‌شود.

برای آنکه تفاوت بین «کیفیت خدمات مبتنی بر جریان» و «کیفیت خدمات مبتنی بر کلاس» روشنتر شود نمونه‌ای مثل «تلفن اینترنتی» را مدنظر قرار بدهید. در روش مبتنی بر جریان، هر تماس تلفنی منابع خاصّ خود و تضمینهای لازم را شبکه اخذ می‌کند. در روش مبتنی بر کلاس تمام تماسهای تلفنی همگی از منابع رزرو شده‌ای که برای «کلاس تلفنی» تهیه دیده شده، استفاده می‌کنند. این منابع در اختیار بسته هایی که در کلاس انتقال فایل یا کلاسهای دیگر هستند، قرار نمی‌گیرد و صرفاً برای «کلاس تلفنی» پیش‌بینی شده است ولی اینگونه هم نیست که برای هر تماس تلفنی منابع اختصاصی و مجزا در نظر گرفته شود.

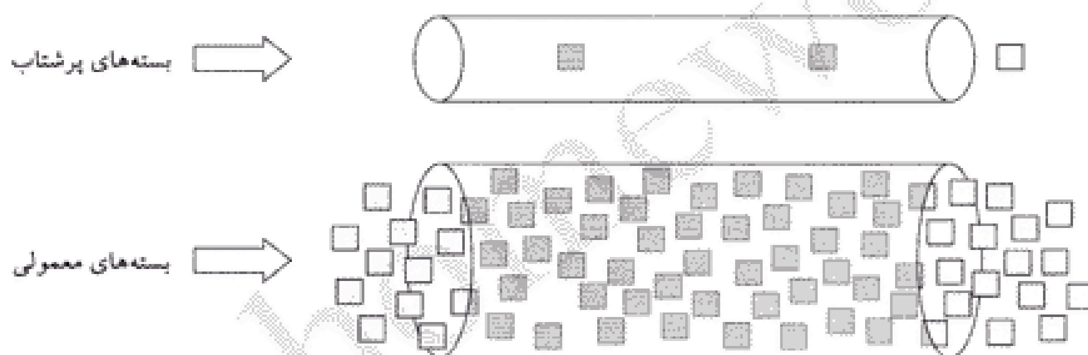
---

۱. Premium Service

هدایت پُرشتاب (Expedited Forwarding)

انتخاب کلاس خدمات بر عهدهٔ کارفرمای شبکه است ولیکن از آنجایی که بسته‌ها از چندین زیرشبکه مجزا (یا کارفرمای مستقل) عبور می‌کنند [و ممکن است کلاس خدمات هر زیرشبکه متفاوت و سلیقه‌ای باشد]، IETF در حال کار بر روی تعریفی واحد برای کلاسهای خدمات است به گونه‌ای که مستقل از نوع شبکه باشد. ساده‌ترین کلاس، کلاس «هدایت پر شتاب» است که با آن شروع می‌کنیم. این کلاس در RFC 3246 تشریح شده است.

ایده‌ای که در پشت روش «هدایت پر شتاب» نهفته است ساده به نظر می‌رسد. خدمات در دو کلاس قابل ارائه است: «معمولی» و «پر شتاب» (Regular & Expedited) بخش اعظم ترافیک شبکه از نوع معمولی هستند در حالی که فقط کسر کوچکی از آن نیاز به «خدمات پُرشتاب» دارند. بسته‌های پُرشتاب باید بگونه‌ای در زیرشبکه حرکت کنند که گویی هیچ بستهٔ دیگری وجود ندارد. توصیفی نمادین از مفهوم سیستم «دو تونلی» (Two-Tube) در شکل ۵-۳۹ ارائه شده است. به خاطر داشته باشید که کماکان یک خط فیزیکی در اختیار است؛ دو لولهٔ منطقی که در شکل نشان داده شده فقط نماد رزرو بخشی از پهنای باند هستند نه آنکه یک خط فیزیکی دیگر هم وجود داشته باشد.



شکل ۵-۳۹. بسته‌های پرشتاب با شبکه‌ای بدون ترافیک مواجه می‌شوند.

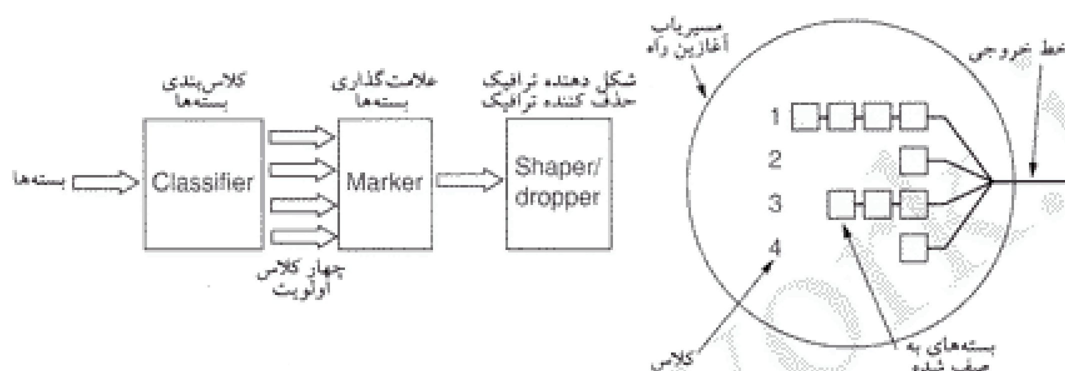
یکی از روشهای پیاده سازی این استراتژی آن است که مسیریابها به نحوی برنامه‌ریزی شوند که برای هر یک از خطوط خروجی خود دو صف مجزا تشکیل بدهند: یکی برای بسته‌های پر شتاب و دیگری برای بسته‌های معمولی. بسته‌های دریافتی برحسب نوع آنها به یکی از این صفها وارد می‌شوند. برای زمان‌بندی بسته‌ها می‌توان از روشی مثل «روش صف‌بندی وزن‌دار» (Weighted Queuing) بهره گرفت. مثلاً اگر ده درصد از بسته‌ها از نوع پر شتاب و مابقی از نوع معمولی باشند، تخصیص ۲۰ درصد از پهنای باند برای ترافیک پر شتاب و هشتاد درصد باقیمانده برای ترافیک معمولی مناسب خواهد بود. با این کار پهنای باند اختصاص داده شده به ترافیک پر شتاب دو برابر مقدار مورد نیاز آن است و بدین ترتیب تأخیر پائینی خواهد داشت. برای اجرای چنین راهکاری می‌توان به ازای ارسال ۴ بستهٔ معمولی یک بستهٔ پر شتاب ارسال کرد (البته با فرض آن که اندازهٔ بسته‌ها توزیعی مشابه و یکنواخت داشته باشد.) انتظار می‌رود در این روش حتی در صورت سنگین بودن بار زیرشبکه، بسته‌های پر شتاب زیرشبکه را بی‌بار و خلوت ببینند.

هدایت تضمین شده (Assured Forwarding)

برای مدیریت انواع کلاسهای خدمات، روشی دقیقتر به نام «هدایت تضمین شده» ارائه و در RFC 2597 تشریح شده است. در این روش چهار کلاس اولویت تعریف شده و هر کلاس منابع خاص خود را در اختیار دارد. علاوه بر این، سه احتمال برای حذف بسته در اثر بروز ازدحام تعریف شده است: احتمال پایین، متوسط و زیاد. مجموع

ترکیبات مختلف این دو عامل، دوازده کلاس خدمات متفاوت ایجاد می‌کند.

در شکل ۵-۴۰ یک روش برای پردازش بسته‌ها به منظور هدایت تضمینی آنها، نشان داده شده است. در مرحلهٔ اول بسته‌ها برحسب کلاس اولویتشان در یکی از چهار کلاس، رده‌بندی می‌شوند. این مرحله می‌تواند در ماشین میزبان فرستندهٔ بسته‌ها انجام شود (به نحوی که در شکل نشان داده شده است) یا آنکه در اولین مسیریاب (مدخل ورودی به زیرشبکه) انجام شود.



شکل ۵-۴۰. پیاده‌سازی مکانیزم هدایت تضمین شده برای یک «جریان داده».

در مرحلهٔ ۲ بسته‌ها بر حسب کلاسشان علامتگذاری می‌شوند. بدین منظور در سرآیند هر بسته به فیلد خاصی نیاز است. خوشبختانه یک فیلد هشت بیتی به نام Type of Service (نوع خدمات) در بستهٔ IP وجود دارد که در آینده آن را به اختصار بررسی خواهیم کرد. در RFC 2597 شش بیت از این هشت بیت برای تعیین کلاس بسته‌ها تعریف شده و دو بیت باقیمانده برای استفاده‌هایی که از قبل داشته یا استفاده در آینده، رها شده‌اند.

در مرحلهٔ سوم بسته‌ها از یک «فیلتر شکل دهنده / حذف کننده» (Shaper / Dropper Filter) عبور کرده و برای آنکه ترافیک بسته‌های هر یک از چهار کلاس شکل قابل قبولی داشته باشند به برخی از آنها تأخیر مصنوعی تحمیل می‌شود (مثلاً به کمک الگوریتم سطل سوراخ یا سطل نشانه‌دار). در صورتی که تعداد بسته‌ها در هر کلاس، از حدّ مجاز بیشتر شده باشد در این مرحله برخی از آنها حذف می‌گردند. (روشهای دقیقتری نیز برای حذف بسته‌ها وجود دارد که از فیدبک بهره می‌گیرند.)

در این مثال هر سه مرحلهٔ فوق‌الذکر توسط ماشین فرستنده انجام شده و جریان خروجی بسته‌ها به اولین مسیریاب ارسال می‌شود. بدیهی است که این مراحل می‌تواند توسط یک نرم‌افزار خاص شبکه یا سیستم عامل هر ماشین انجام شود تا نیازی به تغییر در برنامه‌های کاربردی موجود نباشد.

۵-٤-۵ سوئیچ برچسب[1] و MPLS

در خلال زمانی که IETF بر روی موضوع «خدمات مجتمع» کار می‌کرد، چندین تولیدکنندهٔ محصولات مسیریابی نیز بر روی روشهای بهتر هدایت بسته‌ها متمرکز شده بودند. کار آنها بر این محور بود که در ابتدای هر بسته یک «برچسب» (Label) اضافه شود و بجای آنکه مسیریابی و هدایت بسته‌ها مبتنی بر آدرس مقصد باشد براساس این «برچسب» انجام شود. با استفاده از این «برحسب» به عنوان یک اندیس در جدول داخلی هر مسیریاب، خط خروجی صحیح و مناسب برای هر بسته پیدا می‌شود. بکمک این روش، مسیریابی بسته‌ها به سرعت انجام شده و

---

۱. (Label Switching)

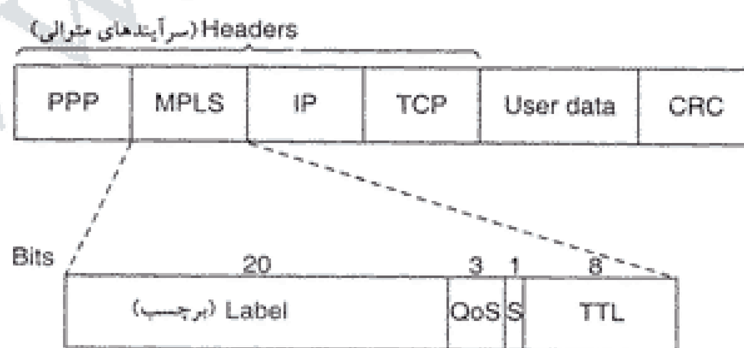
منابع مورد نیاز در طول مسیر رزرو خواهد شد.

البته برچسب‌گذاری بر روی هر «جریان» شباهت عجیبی به مدارهای مجازی پیدا می‌کند. در شبکه‌های ATM، X.25 و Frame Relay یا هر زیرشبکه مدار مجازی دیگر نیز یک «برچسب» (یا به عبارتی یک شناسهٔ مدار مجازی[1]) در هر بسته قرار داده می‌شود و با استفاده از آن به عنوان یک اندیس برای درایه‌های جدول[2]، مسیر مناسب بدست می‌آید. علیرغم آنکه بسیار از افراد در جامعهٔ اینترنت از شبکه‌های اتصال‌گرا بشدت گریزان هستند، به نظر می‌رسد که این ایده با هدف مسیریابی سریع و تأمین کیفیت خدمات (QoS) بار دیگر به صحنه برگشته است. ولیکن بین روشی که در اینترنت برای تعیین مسیر بکار می‌رود و روشی که در شبکه‌های مدار مجازی اعمال می‌شود تفاوتهای بنیادی وجود دارد و تکنیک برچسب گذاری بسته‌ها با روش سنتی سوئیچینگ متفاوت است.

ایدهٔ جدید سوئیچینگ با نامهای متنوعی مثل **«سوئیچینگ برچسب»**[3] یا **«سوئیچینگ علامت»**[4] شناخته می‌شود. در نهایت IETF آن را تحت نام MPLS[5] استاندارد کرد. ما نیز در ادامه از نام MPLS استفاده می‌کنیم. این استاندارد در RFC 3031 و چندین RFC دیگر تشریح شده است.

مضاف بر این، برخی افراد بین «مسیریابی» و «سوئیچینگ» فرق می‌گذارند. مسیریابی فرآیند جستجو در جدول مسیریابی به دنبال آدرس مقصد هر بسته و پیدا کردن خط مناسب برای آن است. برعکس در فرآیند سوئیچینگ از برچسب هر بسته به عنوان یک اندیس در جدول مسیریابی استفاده می‌شود و با استفاده از این اندیس بلافاصله خط خروجی پیدا می‌شود، بدون آن که نیازی به جستجو باشد. البته این تعاریف و تعابیر جهان‌شمول و همگانی نیستند.

اولین مسئله آنست که این برچسب در کجا قرار داده شود. از آنجایی که بسته‌های IP برای شبکه‌های مدار مجازی طراحی نشده بودند، طبعاً هیچ فیلدی در سرآیند بستهٔ IP برای درج شماره‌های مدار مجازی وجود ندارد. به همین دلیل سرآیند جدید MPLS، باید در جلوی سرآیند هر بستهٔ IP قرار بگیرد. در خطوط مستقیم بین هر دو مسیریاب که مبتنی بر «فریمینگ PPP» کار می‌کنند ترتیب سرآیندها طبق شکل ۵-۴۱ عبارتند از: سرآیند PPP، سرآیند MPLS، سرآیند IP و نهایتاً سرآیند TCP. در واقع باید MPLS را در لایه ۲/۵ فرض کرد!!!



شکل ۵-۴۱. ارسال یک قطعهٔ TCP (TCP Segment) با استفاده از IP، MPLS، و PPP.

سرآیند عمومی MPLS (MPLS Header) چهار فیلد دارد که مهمترین آنها فیلد Label (فیلد برچسب) است که در آن یک اندیس درج می‌شود. فیلد QoS، کلاس خدمات را مشخص می‌کند. فیلد S بدان منظور تعریف شده که در شبکه‌های سلسله‌مراتبی چندین سرآیند MPLS متوالیاً به بسته اضافه گردد. (این موضوع در زیر تشریح

---

1. Virtual Circuit Identifier
2. Table Entries
3. Label Switching
4. Tag Switching
5. Multi-Protocol Label Switching

شده است.) فیلد TTL زمان حیات بسته را مشخص می‌کند و به ازای هر گام یک واحد از آن کم می‌گردد؛ هر گاه مقدار این فیلد به صفر برسد، بسته حذف می‌شود. این ویژگی بدان منظور مفید است که از حلقهٔ بی‌نهایت که در اثر ناپایداری (عدم همگرایی) جدول مسیریابی بروز می‌کند، اجتناب شود.

از آنجایی که سرآیند MPLS بخشی از بستهٔ لایهٔ شبکه یا فریم لایهٔ پیوند داده‌ها محسوب نمی‌شود لذا MPLS تا حد زیادی مستقل از هر دو لایه است. از بین تمام محاسن دیگر، دستاورد ویژگی «استقلال از دیگر لایه‌ها» آنست که می‌توان سوئیچهای MPLS را به گونه‌ای ساخت که بتواند هم بسته‌های IP و هم سلولهای ATM را برحسب مورد، هدایت کند. این ویژگی همانی است که براساس آن کلمهٔ Multiprotocol در ابتدای نام MPLS ظاهر شده است.

وقتی یک بسته یا سلول غنی‌شده با سرآیند MPLS در یک مسیریاب MPLS دریافت می‌شود از برچسب آن به عنوان اندیسی در جدول داخلی مسیریاب استفاده شده و خط خروجی متناسب با آن تعیین می‌شود و قبل از خروج بسته از آن خط، برچسب جدیدی در فیلد مربوطه درج می‌گردد. تغییر در برچسبها در تمام زیرشبکه‌های مدار مجازی معمول و متعارف است چرا که برچسبها در هر مسیریاب معنای محلی دارند و دو مسیریاب متفاوت ممکن است بسته‌های نامربوط را با برچسبی یکسان برای مسیریاب دیگر بفرستند چراکه این بسته‌ها همگی در بخشی از مسیر مشترکند.[1] به همین دلیل در هر گام برچسبهای بسته قبل از انتقال بر روی خط خروجی به برچسب جدید و معتبر در مسیریاب بعدی نگاشته می‌شود. این مکانیزم را در شکل ۵-۳ مشاهده کردیم. MPLS نیز از روش مشابهی بهره گرفته است.

یکی از تفاوتهای MPLS با شبکه‌های مدار مجازی، «**میزان تجمیع**» (Aggregation Level) و صرفه‌جویی در تعداد درایه‌های جدول[2] مسیریابی است. در MPLS این امکان وجود دارد که هر «جریان» در زیرشبکه، دارای مجموعهٔ برچسبهای خاص خود باشد ولی این قابلیت مهم نیز وجود دارد که گروهی از جریانها که همگی به یک مسیریاب خاص یا یک LAN ختم می‌شوند با برچسب یکسان و واحدی مشخص شوند. [بدین ترتیب تعداد درایه‌های جدول مسیریابی کاهش یافته و اصطلاحاً عمل تجمیع یا Aggregation انجام می‌شود.] گروهی از جریانها که با یک برحسب واحد مشخص می‌شوند اصطلاحاً به یک FEC[3] مشابه متعلق هستند. [یعنی همهٔ آنها به یک طریق هدایت می‌شوند.] کلاس FEC نه تنها مقصد همهٔ بسته‌ها را مشخص می‌کند، بلکه کلاس خدمات مورد نیاز آنها را نیز تعیین می‌نماید (از دیدگاه انواع خدمات متمایز که در بخش ۵-۴-۴ بدان اشاره شد). طبعاً فرآیند هدایت تمام بسته‌های یک FEC یکسان خواهد بود.

در مسیریابی متعارف به روش مدار مجازی، این قابلیت که بتوان چندین مسیر مجزا با نقاط پایانی متفاوت را با «شناسهٔ مدار مجازی» واحد مشخص کرد وجود ندارد چرا که در این صورت راهی برای مشخص کردن مقصد نهایی بسته‌ها وجود نخواهد داشت در حالی که در MPLS بسته‌ها [به غیر از سرآیند ۴ بایتی MPLS] آدرس واقعی ماشین مقصد را نیز با خود حمل می‌کنند و بدین ترتیب در انتهای مسیری که با برچسب مشخص شده می‌توان سرآیند حاوی برچسب را حذف کرد و هدایت بسته‌ها به روش معمول و مبتنی بر آدرس لایهٔ شبکه [مثل آدرس IP] ادامه یابد.

یکی از تفاوتهای بنیادی بین MPLS و شبکه‌های مدار مجازی در چگونگی تشکیل جداول مسیریابی است.

---

۱. بعبارت دیگر برچسب هر بسته هویت آنرا مشخص نمی‌کند بلکه مسیر خروج آن از مسیریاب فعلی را مشخص می‌کند لذا طبیعی است که بسته‌های خروجی از یک مسیریاب که هیچ ربطی بهم ندارند ولی لااقل گام بعدی مسیر آنها یکی است (یعنی از خط مشابهی وارد مسیریاب بعدی و از خط مشابهی، از آن خارج می‌شوند) دارای برچسب یکسانی باشند. -م

۲. Table Entries ۳. Forwarding Equivalence Class

در شبکه‌های مدار مجازی وقتی یک کاربر بخواهد یک «اتصال» ایجاد کند، توسط لایهٔ شبکه یک بستهٔ خاصّ جهت تنظیم مسیر به زیرشبکه روانه می‌شود تا ضمن ایجاد یک مسیر درایه‌های لازم در جداول مسیریابی درج شود. MPLS بدین نحو عمل نمی‌کند چرا که در آن عمداً هیچ مرحله‌ای برای تنظیم اتصال پیش‌بینی نشده است. (زیرا در غیر این صورت نرم‌افزارهای موجود اینترنت دچار شکاف و ناسازگاری می‌شد.) در عوض برای تنظیم و ایجاد درایه‌های جدول مسیریابی از دو راهکار جدید استفاده شده است.

در راهکار اول که **«روش متکی به داده»** (Data driven) نامیده می‌شود هرگاه بسته‌ای در اولین مسیریاب دریافت شود، آن مسیریاب با مسیریاب واقع بر روی مسیر جریان، تماس گرفته و از او می‌خواهد که یک برچسب برای این جریان ایجاد نماید. این فرآیند به صورت بازگشتی (Recursive) ادامه می‌یابد تا مجموعهٔ برچسبها ایجاد شوند. در واقع این روش را می‌توان ایجاد «مدار مجازی برحسب تقاضا»[1] فرض کرد.

پروتکلهایی که عمل برچسب‌دهی را انجام می‌دهند مراقب هستند تا از بروز حلقه اجتناب شود. برای این کار از تکنیکی به نام «ریسمانهای رنگی» (Colored Thread) بهره گرفته می‌شود. انتشار معکوس یک FEC را می‌توان با یک «ریسمان رنگی و یکتا» [تمثیلی از یک مسیر در شبکه] در زیرشبکه مقایسه کرد. اگر مسیریاب رنگی را مشاهده کند که خودش نیز به همان رنگ است متوجه می‌شود که در انتخاب مسیر، حلقه ایجاد شده و برای رفع آن اقدام می‌کند.[2] روش «برچسب دهی متکی به داده» (Data driven) در شبکه‌هایی کاربرد دارد که زیر ساخت انتقال آنها ATM است. (همانند بیشتر سیستمهای تلفن)

راهکار دیگر برای برچسب دهی به جریان داده‌ها، در شبکه‌هایی کاربرد دارد که زیربنای آنها ATM نیست. این روش اصطلاحاً **«روش متکی به کنترل»** (Control Driven) نامیده می‌شود و گونه‌های متنوعی از آن وجود دارد. یکی از این گونه‌ها به ترتیب ذیل عمل می‌کند: وقتی یک مسیریاب راه‌اندازی (بوت) می‌شود ابتدا بررسی می‌کند که در انتهای چه مسیرهایی قرار دارد (یعنی مثلاً چه ماشینهایی بر روی LAN متصل به او قرار دارند). سپس برای تمام آنها یک یا چند FEC [شناسهٔ یک گروه با کلاس معادل] تولید کرده و ضمن تخصیص یک برچسب به هر یک از این گروهها، آنها را به همسایه‌های خود اطلاع می‌دهد. آنها نیز به ترتیب برچسبها را در جدول مسیریابی خود وارد کرده و با تعیین برچسبی جدید [متناظر با هر برچسب قبلی] آنها را به همسایه‌های خود اطلاع می‌دهند تا آنکه تمام مسیریابها از مسیرهای جدید آگاه شوند. در حین ایجاد مسیر می‌توان منابع لازم را نیز برای تضمین کیفیت خدمات رزرو کرد.

MPLS می‌تواند بطور همزمان در چندین سطح عمل کند. در بالاترین سطح ، هر زیرشبکه حامل را می‌توان یک نوع Metarouter فرض کرد که بین هر مبداء و مقصد مسیری وجود دارد که از این **متاروترها** می‌گذرد؛ در این مسیر از MPLS استفاده می‌شود.[3] با این حال در درون یک زیرشبکه حامل نیز می‌توان از MPLS بهره گرفت و بدین ترتیب مسیریابهای داخلی نیز برچسب دومی به هر بسته می‌افزایند و برچسب گذاری سطح دوم پدید می‌آید. در حقیقت یک بسته می‌تواند دنباله‌ای از برچسبهای MPLS را به همراه داشته باشد. بیت S در شکل ۵-۴۱ مسیریاب را آگاه می‌کند که آیا برچسبهای دیگری هم وجود دارد. در آخرین برچسب، بیت S یک است؛ در حالیکه در بقیهٔ برچسبها بیت S صفر می‌باشد. در عمل می‌توان از این قابلیت برای پیاده‌سازی VPN یا تونلهای بازگشتی

---

۱. On-Demand Vitual Circuit

۲. به عبارت بهتر اگر یک مسیر بین دو نقطه را در قالب یک ریسمان رنگی مجسم کنیم هر بسته‌ای که با رنگ همان مسیر دو بار دریافت شود نشان می‌دهد که بسته در یک حلقه قرار گرفته است وگرنه باید تا رسیدن به مقصد راه خود را ادامه بدهد. -م

۳. یعنی مسیر بین مبداء و مقصد از زیرشبکه‌های متفاوتی می‌گذرند که هر یک از این زیرشبکه‌های حامل یک مسیریاب واحد به نام «متاروتر» فرض شده است. -م

(Recursive Tunnel) بهره گرفت.

اگرچه ایدهٔ بنیادی MPLS ساده است ولیکن جزئیات آن بی‌نهایت پیچیده است و تنوع و بهینه‌سازیهای گسترده‌ای دارد، لذا ما بیش از این به موضوع فوق نخواهیم پرداخت. برای آگاهی بیشتر از مراجع ذیل استفاده کنید:

Davie and Rekhter, 2000; Lin et al., 2002; Pepelnjak and Guichard, 2001; Wang, 2001.

۵-۵ بهم‌بندی شبکه‌ها (Internetworking)

تا اینجا تلویحاً فرض کرده‌ایم که تنها یک شبکهٔ واحد و همگن وجود دارد که تمام ماشینهای چنین شبکه‌ای، در تمام لایه‌ها از پروتکل مشابهی بهره گرفته‌اند. متأسفانه چنین فرضی خیلی خوش‌باورانه است. شبکه‌ها اعم از LAN ، MAN و WAN ، انواع بسیار گوناگونی دارند. در هر لایه نیز از پروتکلهای متعددی استفاده می‌شود. در بخشهای آتی مواردی را موشکافی خواهیم کرد که در وصل دو یا چند شبکه و تشکیل «اینترنت»[۱] (internet) با آن مواجه خواهیم بود.

مناقشات گسترده‌ای پیرامون این سؤال وجود دارد که آیا کثرت بسیار زیاد انواع شبکه‌ها در حال حاضر، وضعیتی گذرا و موقتی است و به محض آنکه عموم مردم به شگفتی‌های یک شبکهٔ خاص (مثلاً شبکهٔ مورد نظر شما!) پی بردند این کثرت به وحدت می‌رسد یا آنکه تکثر انواع امری اجتناب‌ناپذیر و همیشگی در جهان است و باقی خواهد ماند. وجود شبکه‌های متفاوت مستلزم داشتن پروتکلهای متفاوت است.

اعتقاد ما بر آن است که به دلایل ذیل گونه‌های متفاوتی از شبکه‌ها (و به تبع آن پروتکلهای متفاوت) تا ابد وجود خواهد داشت: اول آنکه شبکه‌های بسیار متنوعی در محیطهای متفاوتی نصب شده‌اند: تقریباً تمام کامپیوترهای شخصی بر مبنای TCP/IP کار می‌کنند. بسیاری از مؤسسات تجاری دارای کامپیوترهای بزرگ (Mainframe) با معماری شبکهٔ SNA (متعلق به شرکت IBM) هستند. تعداد قابل توجهی از شرکتهای مخابرات تلفنی، شبکه‌های ATM را به خدمت گرفته‌اند. در برخی از شبکه‌های محلی هنوز از پروتکل NCP/IPX (متعلق به شرکت Novell) یا AppleTalk (متعلق به شرکت Apple) بهره می‌گیرند. و از همه گذشته شبکه‌های بی‌سیم با پروتکلهای متنوعی در حال ظهور هستند. این رویه برای سالها ادامه خواهد داشت چرا که تکنولوژیهای جدید ظهور می‌کند، اشکالات و ناکارآمدیهای گذشته آشکار می‌شود؛ در عین حال با عرضهٔ تکنولوژی جدید نمی‌توان یک شبه مشتریان را وادار کرد سیستمی جدید را بپذیرند و سیستمهای قدیمی خود را دور بریزند.

دلیل دوم آنکه کامپیوترها و شبکه‌ها روز به روز ارزانتر می‌شوند و تصمیم‌گیری در خصوص انتخاب و پیاده‌سازی شبکه‌ها به رده‌های پائینی یک سازمان محول می‌شود. بسیاری از شرکتها این سیاست را پیش گرفته‌اند که خریدهای بالای یک میلیون دلار باید توسط مدیر ارشد آن شرکت تأئید شود و خریدهای بالای صد هزار دلار توسط مدیران میانی مُجاز است در حالی که خریدهای زیر صد هزار دلار توسط مدیران هر واحد بدون اجازهٔ مدیران مافوق انجام می‌گیرد. این رویه به آنجا منتهی می‌شود که مثلاً واحد فنی مهندسی در یک سازمان، ایستگاههایی مبتنی بر یونیکس و پروتکل TCP/IP پیاده کند و واحد فروش، کامپیوترهای Mac با پروتکل AppleTalk را به خدمت بگیرد.

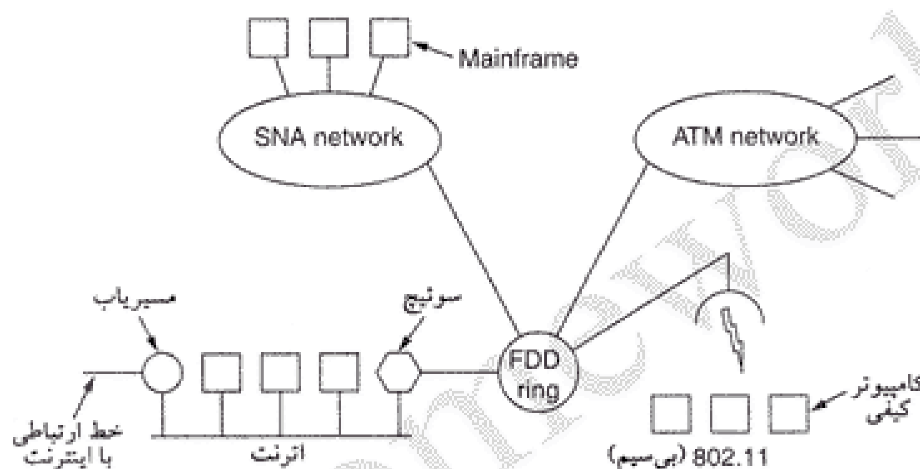
دلیل سوم آنکه شبکه‌های مختلف (مثل ATM و بی‌سیم) تکنولوژی شدیداً متفاوتی دارند لذا دور از انتظار نیست که وقتی سخت‌افزار جدیدی خلق می‌شود نرم‌افزار جدیدی نیز برای به خدمت‌گیری آن ایجاد شود. به

---

۱. وقتی کلمهٔ internet تماماً با حروف کوچک نوشته می‌شود به شبکهٔ عظیم و جهانی اینترنت اشاره نمی‌کند بلکه مراد از آن وصل چند شبکهٔ کوچک و بزرگ و یکپارچه‌سازی آنهاست؛مخفّف internetwork

عنوان مثال امروزه منازلی که در آنها کامپیوتر وجود دارد همانند ادارات در ده سال قبل است: مملو از کامپیوترهایی که ارتباطی با هم ندارند! در آینده ممکن است تلفن، تلویزیون و دستگاههای الکترونیکی خانگی نیز با یکدیگر شبکه شوند و بتوان از راه دور آنها را کنترل کرد. این تکنولوژی جدید بی‌شک شبکه‌های جدید و پروتکلهای جدیدی را به صحنه خواهد آورد.

به عنوان مثالی از چگونگی اتصال شبکه‌های متفاوت به یکدیگر به شکل ۵-۴۲ دقت کنید. در این شکل شبکهٔ یکپارچه‌ای را می‌بینیم که اجزای آن در چندین موقعیت فیزیکی پراکنده هستند و از طریق یک شبکهٔ گستردهٔ ATM بهم متصل شده‌اند. در یکی از مکانها یک شبکهٔ فیبرنوری FDDI به عنوان ستون فقرات نصب شده است تا ارتباط یک شبکهٔ اترنت، یک شبکهٔ بی‌سیم 802.11 و یک شبکه متمرکز SNA را با یکدیگر برقرار کند.



شکل ۵-۴۲. مجموعه‌ای از شبکه‌های بهم متصل.

هدف از اتصال تمام این شبکه‌ها آن است که کاربران هر یک از آنها بتوانند با یکدیگر مبادله اطلاعات داشته باشند یا آنکه هر کاربر بتواند به اطلاعات مورد نظر خود در هر نقطه از شبکه دسترسی پیدا کند. رسیدن بدین هدف متضمن آن است که بسته‌ها، بین این شبکه‌ها رد و بدل شوند. از آنجایی که شبکه‌ها اختلافات بنیانی با یکدیگر دارند، رساندن بسته‌ها از یک شبکه به شبکهٔ دیگر به نحوی که در ادامه خواهیم دید، چندان هم ساده نیست.

۵-۵-۱ شبکه‌ها از چه دیدگاهی متفاوتند؟

شبکه‌ها در موارد مختلفی با یکدیگر تفاوت ذاتی دارند. برخی از این تفاوتها مثل تکنیکهای مدولاسیون یا قالب فریم، مربوط به لایهٔ فیزیکی یا لایهٔ پیوند داده‌ها است. اینگونه تفاوتها مدنظر ما نیستند. در مقابل، در شکل ۵-۴۳ برخی از تفاوتهایی را که در لایهٔ شبکه بروز می‌کنند، فهرست نموده‌ایم. تشریح این تفاوتهاست که نشان می‌دهد بهم بستن شبکه‌ها دشوارتر از کار کردن در یک شبکهٔ واحد و همگون است.

وقتی بستهٔ ارسالی از یک مبداء در یکی از شبکه‌ها، مجبور باشد برای رسیدن به شبکهٔ مقصد از یک یا چند شبکهٔ خارجی عبور کند (که این شبکه‌ها نیز ممکن است با شبکهٔ مبداء اختلاف بنیادی داشته باشند)، در مرز ارتباطی بین دو شبکه مشکلات عدیده‌ای رخ می‌دهد. اولین مورد آن است که وقتی بسته‌هایی از یک شبکهٔ «اتصال‌گرا» مجبور به عبور از یک شبکهٔ «بدون اتصال» باشند (که احتمالاً ترتیب بسته‌ها را بهم می‌ریزد) این مشکل بروز می‌کند که فرستندهٔ بسته‌ها انتظار چنین رخدادی را ندارد و گیرنده نیز کاری نمی‌تواند انجام بدهد. [چرا که فرض فرستنده و گیرنده آن است که بسته‌ها به ترتیب ارسال و به ترتیب نیز دریافت می‌شود. -م]

غالباً بین دو شبکه به تبدیل پروتکل نیاز است و اگر عملکرد مورد نیاز برآورده نشود این تبدیل دشواریهایی را در

مورد اختلاف	برخی از رویکردهای ممکن
نوع سرویس ارائه شده	سرویسهای اتصال‌گرا در مقابل سرویسهای بدون اتصال
انواع پروتکل	IP, IPX, SNA, ATM, MPLS, AppleTalk, ...
الگوی آدرس‌دهی	روش مسطح (Flat) مثلا در استانداردهای ۸۰۲ در مقابل روش سلسله‌مراتبی در IP
چندپخشی	در برخی از شبکه‌ها از آن پشتیبانی می‌شود و در برخی نمی‌شود.
اندازه بسته	هر شبکه برای خودش یک سقف حداکثر برای طول بسته تعریف کرده است.
کیفیت خدمات (QoS)	در برخی از شبکه‌ها از آن پشتیبانی می‌شود (آنهم در رده‌های متفاوت) و در برخی نمی‌شود.
مدیریت خطا	تحویل مطمئن و به ترتیب در مقابل تحویل غیرقابل اطمینان و خارج از ترتیب
کنترل جریان	پنجره لغزان، کنترل نرخ ارسال یا حتی بدون مکانیزم کنترل جریان
کنترل ازدحام	اعمال الگوریتم سطل سوراخ، الگوریتم سطل نشانه‌دار، بسته‌های دعوت به آرامش و نظائر آن
امنیت	قوانین امنیتی، اعمال روشهای رمزنگاری و نظائر آن
پارامترها	مقادیر مختلف زمان انقضای مهلت تایمرها، پارامترهای متفاوت توصیف جریان و نظائر آن
حسابرسی و دریافت هزینه	بر حسب زمان اتصال، برحسب تعداد بسته، بایت یا حتی هیچکدام

شکل ۵-۴۳. موارد بی‌شمار اختلاف شبکه‌ها.

پی خواهد داشت. همچنین اغلب به تبدیل و نگاشت آدرسها نیاز است که متضمن وجود گونه‌ای از یک «سیستم فهرست» (Directory System) خواهد بود. از طرفی عبور یک بستهٔ چندپخشی (Multicast) از شبکه‌ای که نمی‌تواند از مسیریابی چندپخشی پشتیبانی کند مستلزم تولید بسته‌های جداگانه برای یکایک ماشینهای مقصد است.

تفاوت در مقدار حداکثر طول هر بستهٔ داده در شبکه‌های مختلف، می‌تواند یک معضل اساسی باشد: چگونه می‌توان یک بستهٔ ۸۰۰۰ بایتی را از شبکه‌ای عبور داد که حداکثر طول بسته‌های آن ۱۵۰۰ بایت است؟ تفاوت در کیفیت خدمات دو شبکه (QoS) نیز موردی است که برای تحویل بسته‌های بی‌درنگ از طریق شبکه‌ای که بی‌درنگ بودن ارسال را تضمین نمی‌کند، به یک معضل جدی تبدیل می‌شود.

کنترل خطا، کنترل جریان و کنترل ازدحام نیز در شبکه‌های مختلف، تفاوت دارد. اگر مبداء و مقصد، هر دو انتظار داشته باشند که تمام بسته‌ها به ترتیب و بدون خطا تحویل شوند ولی یک شبکهٔ میانی به محض احساس بروز ازدحام برخی از بسته‌ها را حذف نماید، بسیاری از برنامه‌های کاربردی درهم خواهند شکست. همچنین اگر بسته‌ها مدتی را بی‌هدف سرگردان شوند و به ناگاه راه خود را پیدا کرده و تحویل داده شوند و گیرنده انتظار چنین رفتاری را نداشته و از عهدهٔ دریافت این بسته‌ها بر نیاید معضلی جدی پدیدار می‌شود. همچنین مکانیزمهای تضمین امنیت، تنظیم پارامترها، قواعد دریافت هزینه (حسابرسی) و حتی قوانین ملی متفاوت، می‌تواند منجر به بروز مشکلاتی شود.

۵-۵-۲ چگونگی اتصال شبکه‌ها به یکدیگر

به گونه‌ای که در فصل چهارم بررسی کردیم شبکه‌ها را می‌توان به کمک ابزارهای متفاوتی به یکدیگر متصل کرد. اجازه بدهید اجمالاً به آن مفاد بپردازیم: در لایهٔ فیزیکی شبکه‌ها را می‌توان توسط «تکرارکننده» (Repeater) یا هاب به یکدیگر متصل کرد. این دو ابزار فقط بیتها را از یک شبکه به شبکه‌ای از همان نوع منتقل می‌نمایند. اینها اغلب ابزارهای آنالوگ هستند و هیچ درکی در خصوص پروتکلهای دیجیتال [پروتکلهای لایهٔ بالاتر] ندارند و فقط سیگنالهای دریافتی را «باز تولید» (Regenerate) می‌نمایند.

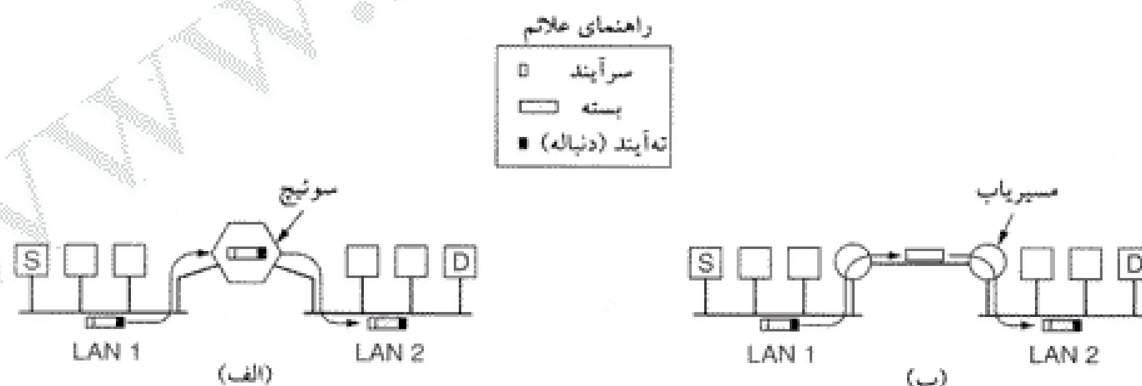
در یک لایهٔ بالاتر به پلها و سوئیچها بر می‌خوریم که در لایهٔ پیوند داده‌ها کار می‌کنند. این دستگاهها فریمها را می‌پذیرند، آدرسهای MAC را بررسی می‌کنند و آنها را به شبکه‌ای دیگر هدایت می‌نمایند. در ضمن اگر لازم باشد تبدیل پروتکل نیز انجام می‌دهند مثلاً اترنت را به FDDI یا 802.11 تبدیل می‌کنند.

در لایهٔ شبکه، مسیریابها را داریم که می‌توانند دو شبکه را به یکدیگر متصل کنند. اگر دو شبکه، لایه‌های شبکهٔ متفاوتی داشته باشند، مسیریاب ممکن است بتواند قالب بسته‌ها را به یکدیگر ترجمه نماید اگرچه امروزه ترجمه و تبدیل بسته‌ها به یکدیگر، به ندرت انجام می‌شود. یک مسیریاب را که بتواند با چندین پروتکل مختلف کار کند، اصطلاحاً «مسیریاب چندپروتکلی» (Multiprotocol Router) می‌نامند.

در لایهٔ انتقال به «دروازه‌های انتقال» (Transport Gateway) می‌رسیم که می‌توانند واسطهٔ بین دو اتصال در لایهٔ انتقال شوند. به عنوان مثال یک «دروازهٔ انتقال» می‌تواند این امکان را فراهم کند که جریان بسته‌ها بین یک شبکهٔ TCP و یک شبکهٔ SNA (که پروتکل لایهٔ انتقال آنها متفاوت است)، مبادله شود. این دروازه، یک «اتصال TCP» (TCP connection) را به یک «اتصال SNA» می‌چسباند.

در آخر به لایهٔ کاربرد و «دروازهٔ کاربرد» (Application Gateway) می‌رسیم که این دروازه محتوای پیامها را بهم ترجمه می‌نماید. به عنوان مثال یک دروازهٔ بین سیستم پست الکترونیکی در اینترنت (مثل سیستم RFC822) و سیستم پست الکترونیکی X.400، باید بتواند محتوای پیام نامه‌های الکترونیکی را تجزیه و تحلیل کرده و فیلدهای مختلف سرآیند آنرا به یکدیگر تبدیل نماید.

در این فصل به موضوع بهم‌بندی شبکه‌ها در لایهٔ شبکه خواهیم پرداخت. برای آنکه ببینید هدایت اطلاعات در لایهٔ پیوند داده‌ها چه تفاوتی با هدایت در لایهٔ شبکه دارد، شکل ۵-۴۴ را در نظر بگیرید. در شکل ۵-۴۴-الف ماشین مبداء یعنی S می‌خواهد بسته‌ای را برای ماشین مقصد D بفرستد. این دو ماشین بر روی دو شبکهٔ اترنت جدا که از طریق سوئیچ بهم متصل شده‌اند، واقع هستند. ماشین S بسته‌ای را در درون یک فریم جاسازی کرده و آن را به خروجی می‌فرستد. این فریم به سوئیچ رسیده و با بررسی آدرس MAC مشخص می‌شود که باید به LAN2 برود. سوئیچ، این فریم را از LAN1 برداشته و به LAN2 روانه می‌کند.



شکل ۵-۴۴. (الف) دو شبکهٔ اترنت که از طریق سوئیچ بهم متصل شده‌اند. (ب) دو شبکهٔ اترنت که از طریق مسیریاب بهم متصل شده‌اند.

حال همین وضعیت را در شرایطی در نظر بگیرید که این دو شبکهٔ اترنت به جای سوئیچ از طریق یک جفت مسیریاب به یکدیگر متصل شده‌اند. ارتباط بین مسیریابها از طریق یک خط نقطه‌به‌نقطه برقرار شده است؛ این خط می‌تواند یک خط استیجاری با هزاران کیلومتر طول باشد. در اینجا فریم توسط مسیریاب اول دریافت شده و بستهٔ جاسازی شده در درون آن، از فیلد دادهٔ فریم استخراج می‌شود. مسیریاب، آدرس درون بسته را (مثلاً آدرس IP

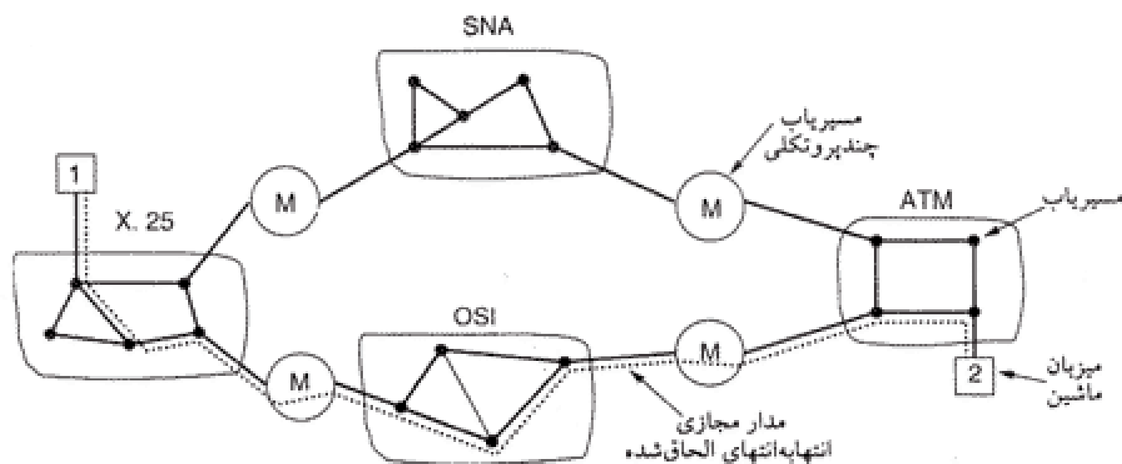
را) بررسی کرده و آنرا در درون جدول مسیریابی خود جستجو می‌نماید؛ سپس براساس این آدرس به نتیجه می‌رسد که باید بسته را به مسیریاب راه دور بفرستد و احتمالاً برای این کار مجبور خواهد شد که مبتنی بر پروتکل این خط ، آنرا در فریم جدید و متفاوتی جاسازی نماید. در مسیریاب مقابل این بسته مجدداً درون فیلد داده از یک فریم اترنت جاسازی شده و بر روی LAN2 ارسال می‌شود.

تفاوت بنیادی بین حالتی که سوئیچ (یا پل) در میان است با وقتی که مسیریاب اتصال شبکه را برقرار کرده ، آنست که در سوئیچ یا پل کلّ یک فریم براساس آدرس MAC آن هدایت و منتقل می‌شود در حالیکه در یک مسیریاب، یک بسته از درون فریم استخراج شده و سپس از آدرس درون بسته برای تصمیم‌گیری در خصوص محل ارسال آن استفاده می‌شود. سوئیچها مجبور نیستند که برای هدایت بسته‌ها درکی از پروتکل لایهٔ شبکه داشته باشند در حالیکه مسیریابها اینگونه‌اند.

۵-۵-۳ مدارات مجازی الحاق‌شده (Concatenated Virtual Circuit)

دو روش برای بهم بندی شبکه‌ها ممکن است: روش الحاق زیرشبکه‌های مدار مجازی به صورت اتصال‌گرا و روش الحاق دیتاگرام. به نوبت این دو روش را بررسی خواهیم کرد ولی در ابتدا به ذکر نکته‌ای می‌پردازیم. در گذشته اکثر شبکه‌های عمومی اتصالگرا بودند (و شبکه‌هایی مثل Frame Relay، SNA ، 802.16 و ATM هنوز هم اینگونه‌اند). با رشد و مقبولیت سریع اینترنت، شبکهٔ دیتاگرام مُد روز شد ولیکن این تصور که شبکه‌های دیتاگرام ابدی هستند، اشتباه است. در دنیای شبکه‌ها تنها چیزی که جاوید و ابدی می‌ماند «**تغییر**» است. با رشد شبکه‌های چند رسانه‌ای، احتمالاً اتصال‌گرایی با یکی از آشکال خود مجدداً به صحنه برمی‌گردد چراکه در شبکه‌های اتصال‌گرا راحتتر می‌توان کیفیت خدمات را تضمین نمود. لذا در ابتدا شبکه‌های اتصال‌گرا را مورد بحث و بررسی قرار می‌دهیم.

در مدل الحاق شبکه‌های مدار مجازی که در شکل ۵-۴۵ نشان داده شده، ایجاد «اتصال» با یک ماشین در شبکه‌ای دوردست، بروشی مشابه با روش معمولی انجام می‌گیرد: زیرشبکه می‌بیند که مقصد در شبکه‌ای دیگر واقع شده لذا یک مدار مجازی با آن مسیریاب که به شبکهٔ مقصد نزدیکتر است، ایجاد می‌کند. از آن مسیریاب نیز یک مدار مجازی با یک «دروازهٔ خارجی» ایجاد می‌شود. (دروازهٔ خارجی یا External Router، یک مسیریاب چند پروتکلی است.) آن «دروازه» نیز مشخصات این مدار مجازی را در جدول خود درج کرده و کار را با ایجاد یک مدار مجازی جدید با مسیریاب زیرشبکهٔ دیگر ادامه می‌دهد. این فرآیند ادامه می‌یابد تا آنکه مدار مجازی به ماشین مقصد ختم شود.



شکل ۵-۴۵. بهم‌بندی شبکه‌ها به کمک مدارات مجازی الحاق‌شده.

هرگاه یک بستهٔ داده، در مسیری جریان یابد، هر یک از دروازه‌های میانی این بسته را رله می‌کنند و در صورت نیاز قالب بسته را تبدیل نموده و شماره‌های مدار مجازی را تغییر می‌دهند. بدیهی است که تمام بسته‌های داده باید به یک ترتیب از دروازه‌ها بگذرند؛ نتیجتاً ترتیب بسته‌های متعلق به یک جریان هرگز به هم نخواهد ریخت.

ویژگی بنیادی این راهکار آنست که دنباله‌ای از مدارات مجازی بین ماشین مبداء و مقصد (از طریق دروازه‌های میانی) ایجاد و تنظیم می‌شود. هر دروازه جدولی را در خود نگاهداری می‌کند که این جدول فهرست مدارات مجازی را که از آن دروازه می‌گذرند و همچنین طریقهٔ مسیریابی و شماره‌های جدید مدار مجازی را تعیین کرده است.

این ساختار زمانی به بهترین نحو کار می‌کند که تمام شبکه‌های میانی از ویژگیهای مشابهی برخوردار باشند. به عنوان مثال اگر تمام آنها تحویل مطمئن بسته‌های لایهٔ شبکه را تضمین کرده باشند، آنگاه جریان بین مبداء و مقصد، مطمئن و قابل اعتماد خواهد بود (به استثنای وقتی که یکی از مسیریابهای میانی از کار بیفتد.) به دلیل مشابه اگر هیچیک از شبکه‌های میانی تحویل مطمئن بسته‌ها را تضمین نکرده باشند، الحاق مدارات مجازی نیز نامطمئن خواهد بود. ولیکن اگر ماشین مبداء بر روی شبکه‌ای باشد که تحویل مطمئن بسته‌ها را تضمین کرده در حالی که یکی از شبکه‌های میانی استعداد از بین بردن بسته‌ای را داشته باشد، الحاق مدارات مجازی منجر به نامطمئن شدن کل مدار مجازی شده و طبیعت خدمات پیش‌بینی شده مثل تحویل مطمئن و حفظ ترتیب بسته‌ها تغییر می‌کند.

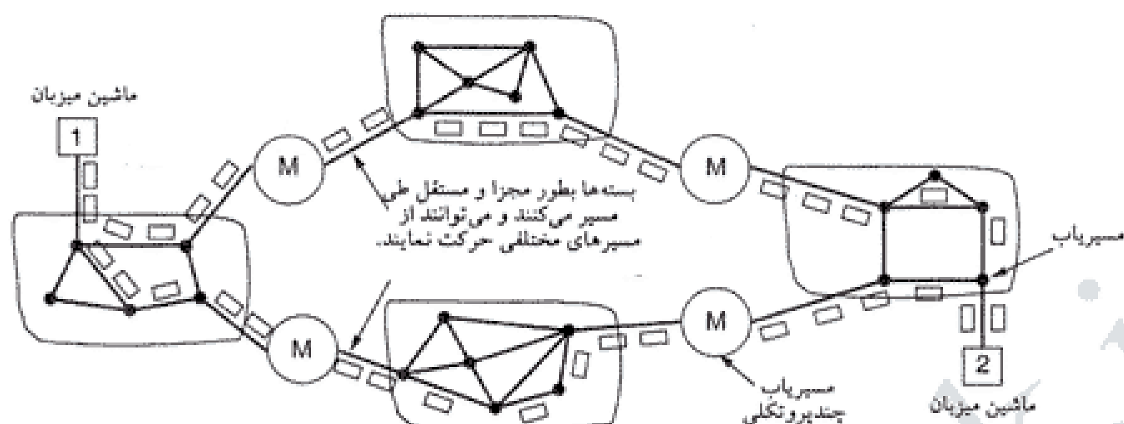
الحاق مدارات مجازی در لایهٔ انتقال نیز رایج است. بویژه این امکان وجود دارد که یک «خط انتقال بیت» (Bit Pipe) بین شبکه‌ای مثل SNA که به یک دروازه منتهی می‌شود و اتصالی TCP با دروازهٔ دیگر دارد، ایجاد نمود. بدین ترتیب می‌توان یک مدار مجازی «انتها به انتها» (End To End) ایجاد کرد در حالی که چندین شبکه با پروتکلهای مختلف در میانهٔ راه قرار گرفته‌اند.

۵-۵-۴ بهم‌بندی شبکه‌های بدون‌اتصال (Connectionless Internetworking)

مدلی دیگر از بهم‌بندی شبکه‌ها ، مدل دیتاگرام است؛ (به شکل ۵-۴۶ دقت کنید). در این مدل تنها خدمتی که لایهٔ شبکه به لایهٔ انتقال ارائه می‌دهد آن است که بسته‌های دیتاگرام را بر روی زیرشبکه تزریق کند؛ زیرشبکه نیز حداکثر تلاش خود را در جهت تحویل آن به عمل می‌آورد. در این مدل هیچگونه نشانی از مدار مجازی در سطح لایهٔ شبکه وجود ندارد و فقط شبکه‌ها به هم متصل و ملحق می‌شوند. در این مدل نیازی نیست که بسته‌های متعلق به یک اتصال [تولید شده توسط یک ماشین] به ترتیب از دروازه‌های یکسانی بگذرند. در شکل ۵-۴۶ دیتاگرامهای ارسالی توسط ماشین ۱ که به سوی ماشین ۲ روانه شده‌اند از مسیرهای متفاوتی در شبکه عبور کرده‌اند. تصمیم‌گیری در خصوص مسیر هر بسته، بطور جداگانه انجام می‌شود و این تصمیم‌گیری بستگی به ترافیک لحظهٔ ارسال بسته دارد. در این استراتژی از چندین مسیر بهره گرفته می‌شود و طبعاً پهنای باند بیشتری در مقایسه با مدل الحاق شبکه‌های مدار مجازی حاصل خواهد شد. ولی در مقابل تضمینی در به ترتیب رسیدن بسته‌ها به مقصد وجود ندارد، بلکه فقط فرض بر تحویل بسته‌هاست نه حفظ ترتیب آنها.

مدل شکل ۵-۴۶ به همین سادگی که بنظر می‌رسد نیست. اولین مورد اشکال آنکه، اگر هر یک از شبکه‌های میانی، پروتکل لایهٔ شبکهٔ خاص خودشان را داشته باشد، انتقال بسته از یک شبکه به شبکه‌ای دیگر ممکن نخواهد بود. شاید کسی تصور کند که یک مسیریاب چند پروتکلی قادر به ترجمهٔ قالب بسته‌ها به یکدیگر است در حالی که این تصور زمانی درست است که قالب بسته‌ها نزدیک به یکدیگر بوده و فیلدهای اطلاعاتی هر بسته مشابه باشند و در غیر این صورت چنین تبدیلی ناقص بوده و محکوم به شکست است. به همین دلیل به ندرت تلاش می‌شود چنین تبدیلی انجام گیرد.

دومین مشکل جدی، مسئله آدرس‌دهی است. یک حالت ساده را مدنظر قرار بدهید: یک ماشین بر روی شبکه



شکل ۵-۴۶. بهم‌بندی شبکه‌های بدون‌اتصال.

اینترنت تلاش می‌کند یک بستهٔ IP برای ماشینی بر روی یک شبکهٔ SNA (متصل به اینترنت) بفرستد. آدرسهای IP و SNA متفاوت از هم هستند. آدرسهای IP و SNA باید در هر دو جهت به یکدیگر نگاشته و ترجمه شوند. مضاف بر این، در شبکه‌های متفاوت مفهوم «چیزهایی که قابل آدرس‌دهی هستند» فرق دارد. در IP ماشینهای میزبان (یا در حقیقت کارتهای واسط شبکه) دارای آدرس هستند. در SNA هر **«موجودیت»** (Entity) مثل هر ابزار سخت‌افزاری می‌تواند آدرس داشته باشد. در بهترین حالت هر دروازه باید دارای یک پایگاه اطلاعاتی بوده و بتواند آدرسها را بهم بنگارد (تبدیل کند) ولی همین کار منشاء بروز مشکلات جدی است.

یک نظریهٔ دیگر آن است که یک بستهٔ جهانی و استاندارد طراحی شود و تمام مسیریابها آنرا به رسمیت بشناسند. این راهکار در حقیقت همین IP است که بسته‌های آن به گونه‌ای طراحی شده که هر شبکه‌ای قادر به حمل و هدایت آنهاست. البته ممکن است به نظر برسد IPv4 (پروتکل فعلی اینترنت) نهایتاً تمام ساختارها و پروتکلهای دیگر را از صحنه خارج می‌کند و IPv6 (پروتکل آیندهٔ اینترنت) نیز راه به جایی نمی‌برد و هیچ پروتکل جدیدی ابداع نمی‌شود! ولی تاریخ نشان داده که اینگونه نیست. جلب موافقت عموم افراد برای پذیرش یک قالب واحد بسیار دشوار است چرا که شرکتهای مختلف مصالح و سود خود را در آن می‌بینند که قالب و ساختار اختصاصی و تحت کنترل خود را داشته باشند.

حال بیایید به اختصار مروری بر دو روش شبکه‌بندی داشته باشیم. مدل الحاق شبکه به روش مدار مجازی همان مزایایی را دارد که مدار مجازی در یک زیرشبکه واحد خواهد داشت: یعنی پیشاپیش می‌توان بافرها را از قبل رزرو کرد، ترتیب بسته‌ها حفظ می‌شود، سرآیند کوتاهتری برای بسته‌ها نیاز است و از مشکلاتی که ناشی از تکراری شدن بسته‌هایی که با تأخیر می‌رسند، احتراز می‌شود.

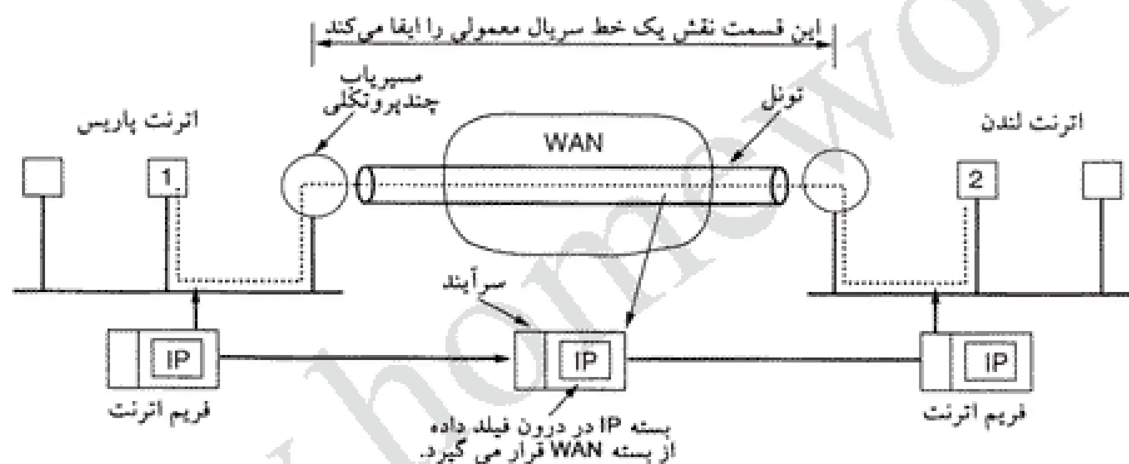
این روش معایبی نیز دارد: در مسیریابها به فضای قابل‌توجهی برای نگاهداری جدول اتصالات باز نیاز است، برای احتراز از مناطق مواجه با ازدحام مکانیزم مسیریابی و مسیرها تغییر نمی‌کند و مسیرها نسبت به خرابی مسیریابها بسیار آسیب‌پذیر هستند. همچنین این اشکال بزرگ وجود دارد که اتصال چنین شبکه‌ای به یک شبکهٔ نامطمئن دیتاگرام اگر غیرممکن نباشد بسیار دشوار است.

ویژگی بهم‌بندی شبکه‌ها به روش دیتاگرام، دقیقاً مشابه با ویژگی زیرشبکه‌های دیتاگرام است: استعداد بروز ازدحام دارد ولی در عوض قابلیت بیشتری نیز برای رفع آن دارد، در مواجهه با خرابی یک مسیریاب قابلیت تحمل بیشتری از خود نشان می‌دهد و به سرآیند طولانی‌تری برای هر بسته نیاز است؛ استفاده از انواع الگوریتمهای وفقی و پویای مسیریابی میسّر است و در مجموع تمام مزایا و معایب یک زیرشبکه واحد و منفرد را بطور مشابه دارد.

بزرگترین مزیت روش دیتاگرام برای بهم‌بندی شبکه آن است که می‌توان این روش را برای وصل هر شبکه‌ای که در درون از مدار مجازی بهره گرفته، استفاده کرد. بسیاری از شبکه‌های LAN ، شبکه‌های متحرک (Mobile) مثل شبکهٔ ناوگانهای هوایی و دریایی و حتی برخی از شبکه‌های WAN ، در ردهٔ شبکه‌های دیتاگرام قرار می‌گیرند و هر گاه استراتژی بهم‌بندی این شبکه‌ها مبتنی بر مدارات مجازی باشد، مشکلات جدی بروز خواهد کرد.

۵ـ۵ـ۵ ایجاد تونل (Tunneling)

اتصال دو شبکهٔ مختلف در حالت کلی بسیار سخت است ولیکن یک حالت خاص و رایج وجود دارد که راهگشا و قابل مدیریت است. این حالت زمانی اتفاق می‌افتد که ماشینهای مبداء و مقصد بر روی شبکه‌هایی از یک نوع هستند ولی یک شبکهٔ متفاوت در میان آنها قرار گرفته است. به عنوان مثال یک بانک بین‌المللی را مدنظر قرار دهید که یک شبکهٔ اترنت مبتنی بر پروتکل TCP/IP در پاریس و همچنین یک اترنت با پروتکل TCP/IP در لندن دارد ولی به نحوی که در شکل ۵–۴۷ نشان داده شده، یک شبکه بغیر از IP (مثل ATM) در میان آنها قرار گرفته است.



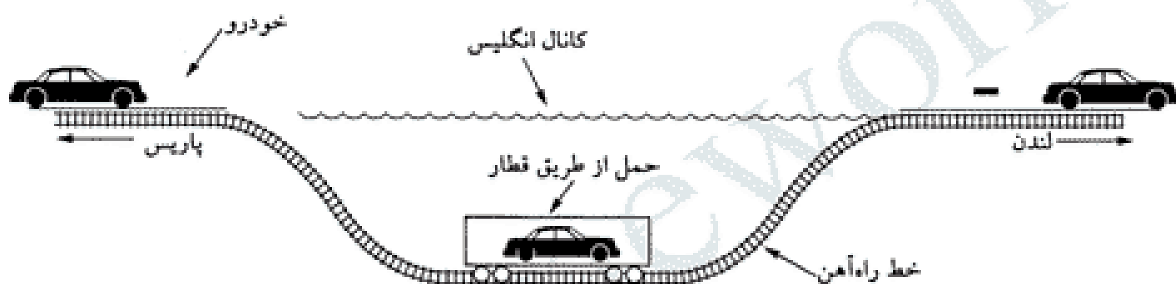
شکل ۵–۴۷. ارسال یک بسته از طریق ایجاد تونل بین پاریس و لندن.

راهکار حل این مشکل تکنیکی به نام «**ایجاد تونل**» است. برای ارسال یک بستهٔ IP به ماشین ۲ ، ماشین ۱ ، بسته‌ای حاوی آدرس IP ماشین ۲ ساخته و آنرا در درون فریم شبکهٔ اترنت قرار داده و آدرس مسیریاب چند پروتکلی واقع در پاریس را در فیلد آدرس این فریم درج کرده و آن را بر روی اترنت قرار می‌دهد. هرگاه این «مسیریاب چندپروتکلی» این فریم را دریافت کند بستهٔ IP را از درون فریم استخراج کرده و آن را در فیلد حمل داده (Payload) از بستهٔ لایهٔ شبکهٔ WAN قرار داده و آدرس بستهٔ جدید را آدرس مسیریاب چندپروتکلی واقع در لندن قرار می‌دهد. [به عبارتی دو بستهٔ تودرتو تشکیل می‌شود که آدرس بستهٔ درونی، آدرس ماشین مقصد و آدرس بستهٔ بیرونی آدرس مسیریاب لندن است.] هر گاه این بسته به مسیریاب واقع در لندن برسد آن مسیریاب بستهٔ IP اصلی را از درون آن استخراج کرده و آنرا با قرار دادن در یک فریم اترنت برای ماشین ۲ می‌فرستد.

شبکهٔ WAN را می‌توان یک تونل بزرگ در نظر گرفت که از یک مسیریاب چندپروتکلی شروع و به مسیریابی دیگر از همین نوع، در طرف دیگر WAN ختم می‌شود. یک بستهٔ IP در حالی که درون یک بستهٔ دیگر جا گرفته از یک طرف تونل شروع به طی مسیر به طرف دیگر تونل می‌کند و جای هیچگونه نگرانی در خصوص ساختار WAN وجود ندارد [چرا که تنها کاری که WAN در این میان انجام می‌دهد آنست که مبتنی بر پروتکل فعلی خود، بستهٔ IP را به عنوان یک قطعه دادهٔ خام، در یک نقطه دریافت و در نقطهٔ دیگر تحویل می‌دهد.-م] در این مثال در خصوص ماشینهای هر یک از شبکه‌های اترنت نیز نگرانی وجود ندارد [چون هر دو از یک نوع و مبتنی بر پروتکل

مشابه هستند]. فقط «مسیریاب چندپروتکلی» است که باید هم بسته‌های IP و هم بسته‌های WAN را بفهمد و مدیریت کند. در حقیقت کل شبکه‌ای که در میان این دو مسیریاب چندپروتکلی قرار گرفته نقشی شبیه به یک خط سریال ایفاء می‌کند.

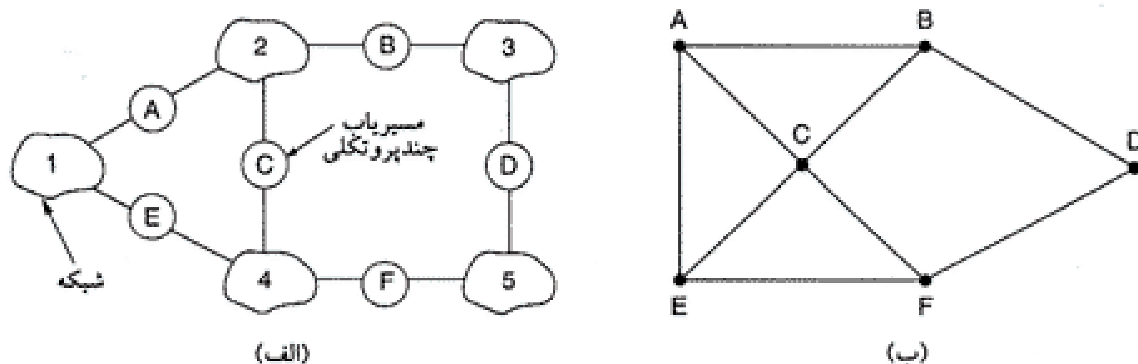
شاید یک تمثیل بتواند مفهوم **تونل** را روشنتر کند. شخصی را در نظر بگیرید که با خودروی خودش از پاریس به لندن مسافرت می‌کند. در فرانسه او با خودروی خود تا کنار کانال انگلیس رانندگی می‌کند ولی با رسیدن به این کانال (که رانندگی در آن ممنوع است)، خودروی او در یک قطار سریع‌السیر بار شده و از طریق قطار به انگلیس منتقل می‌شود. در حقیقت، به نحوی که در شکل ۵-۴۸ نشان داده شده، این خودرو همانند یک محمولهٔ عادی جابجا می‌شود. در طرف دیگر، این خودرو مجدداً بر زمین گذاشته شده و اجازه می‌یابد با نیروی محرکهٔ خودش در جاده‌های انگلیس حرکت نماید. ایجاد تونل برای بسته‌ها از طریق یک شبکهٔ خارجی مشابه با همین عملکرد است.



شکل ۵-۴۸. تونل کردن (Tunneling) یک خودرو از فرانسه به انگلستان.

۵-۵-۶ مسیریابی بین شبکه‌های بهم‌متصل

مسیریابی بین چند شبکهٔ متصل‌بهم مشابه با مسیریابی در یک زیرشبکه واحد است (البته با پیچیدگی‌های بیشتر). به عنوان مثال ساختار ارتباطی شبکهٔ الف-۴۹-۵ را در نظر بگیرید که در آن پنج شبکه توسط شش مسیریاب (احتمالاً از نوع چندپروتکلی) بهم متصل شده‌اند. تشکیل مدل گراف از این وضعیت پیچیده است چراکه در گراف، هر مسیریاب باید مستقیماً با مسیریاب دیگر در ارتباط باشد (بسته بفرستد) در حالیکه اینجا مسیریابها مستقیماً به یک شبکهٔ میانی متصلند. مثلاً مسیریاب B در شکل ۵-۴۹-الف می‌تواند از طریق شبکهٔ ۲ به مسیریابهای A و C دسترسی داشته باشد، همچنین مسیریاب D از طریق شبکهٔ ۳ به B دسترسی دارد. اگر هر یک از شبکه‌ها را به مثابهٔ یک خط سریال فرض نماییم، گراف شکل ۵-۴۹-ب بدست می‌آید.



شکل ۵-۴۹. (الف) «شبکه‌ای از شبکه‌ها» (ب) گراف متناظر.

به محض آنکه گراف شبکه تشکیل شد می‌توان یکی از الگوریتمهای شناخته شدهٔ مسیریابی مثل «الگوریتم بردار فاصله» یا «الگوریتم حالت لینک» (Link State Algorithm) را بر روی مجموعهٔ این مسیریابهای چندپروتکلی اعمال کرد. در نتیجه یک الگوریتم مسیریابی دوسطحی ایجاد می‌شود: در درون هر شبکه از یک «**پروتکل مسیریابی درونی**» (Interior Gateway Protocol) استفاده می‌شود در حالی که بین شبکه‌ها از «**پروتکل مسیریابی بیرونی**» (Exterior Gateway Protocol) بهره گرفته می‌شود. (اصطلاح Gateway نام قدیمی مسیریاب است.) در حقیقت، چون شبکه‌ها مستقل هستند ممکن است از الگوریتمهای مسیریابی مختلفی در درون شبکه بهره گرفته باشند. از آنجایی که با بهم‌بندی شبکه‌ها هر شبکه استقلال داخلی خود را حفظ می‌کند، به هر یک از این شبکه‌های مستقل «**سیستم خودمختار**» یا به اختصار AS گفته می‌شود.

در چنین شبکه‌ای، یک بستهٔ نوعی از یک ماشین بر روی LAN مسیر خود را آغاز کرده و به آدرس «مسیریاب چندپروتکلی» متصل به آن LAN ارسال می‌شود (با قرار دادن آدرس آن مسیریاب در فیلد آدرس از فریم MAC). پس از آن که بسته به مسیریاب رسید، نرم‌افزار آن مسیریاب به کمک جدول مسیریابی خود مشخص می‌کند که این بسته باید به سوی کدامیک از مسیریابهای دیگر هدایت شود. اگر بسته رابتوان با همان پروتکل لایهٔ شبکه (که بسته براساس آن تولید شده) به مقصد رساند، این بسته مستقیماً هدایت و ارسال می‌شود. در غیر این صورت از مکانیزم ایجاد تونل (Tunneling) استفاده شده و کل بسته درون یک بستهٔ متناسب با پروتکل مسیریابهای میانی جاسازی و ارسال می‌گردد. این فرآیند آنقدر تکرار می‌شود تا بسته به شبکهٔ مقصد برسد.

یکی از تفاوتهای بین مسیریابی درون یک شبکهٔ خودمختار (Intranetwork Routing) و مسیریابی بین چند شبکهٔ خودمختار آن است که در مورد دوم ممکن است بسته‌ها نیاز به تردّد از مرزهای بین‌المللی کشورها داشته باشند. در چنین حالتی، ملاحظات قانونی متعدد مطرح خواهد شد چرا که مثلاً طبق قوانین کشور سوئد، خارج کردن اطلاعات فردی شهروندان سوئدی از کشور مجاز نیست. یا مثلاً طبق قوانین کشور کانادا، ترافیک داده‌هایی که مبداء آنها در کشور کانادا و مقصد آنها نیز در کانادا است نباید از کشور خارج شود. طبق این قانون ترافیک داده‌هایی که مبداء آن شهر «ویندزور» یا «انتاریو» و مقصد آن «ونکوور» است نمی‌تواند از طریق مسیر نزدیکتری که از «دیترویت» می‌گذرد، مسیریابی و هدایت شود حتی اگر استفاده از این مسیر سریعتر و ارزانتر باشد.

تفاوت دیگر بین مسیریابی درونی و مسیریابی بیرونی (Interior/Exterior Routing) موضوع هزینه (قیمت) است. در یک شبکهٔ خودمختار و واحد، بطور معمول از یک الگوریتم مشخص برای تعیین هزینه استفاده می‌شود ولیکن در شبکه‌های مختلف که مدیریت متفاوتی دارند ممکن است یک مسیر از لحاظ قیمت از مسیر دیگر مقرون به صرفه‌تر باشد. همینطور سطح کیفیت خدمات عرضه شده در شبکه‌های متفاوت فرق می‌کند و همین مورد ممکن است دلیل انتخاب یک مسیر از بین دیگر مسیرها باشد.

۵-۵-۷ قطعه‌قطعه‌سازی بسته‌ها (Fragmentation)

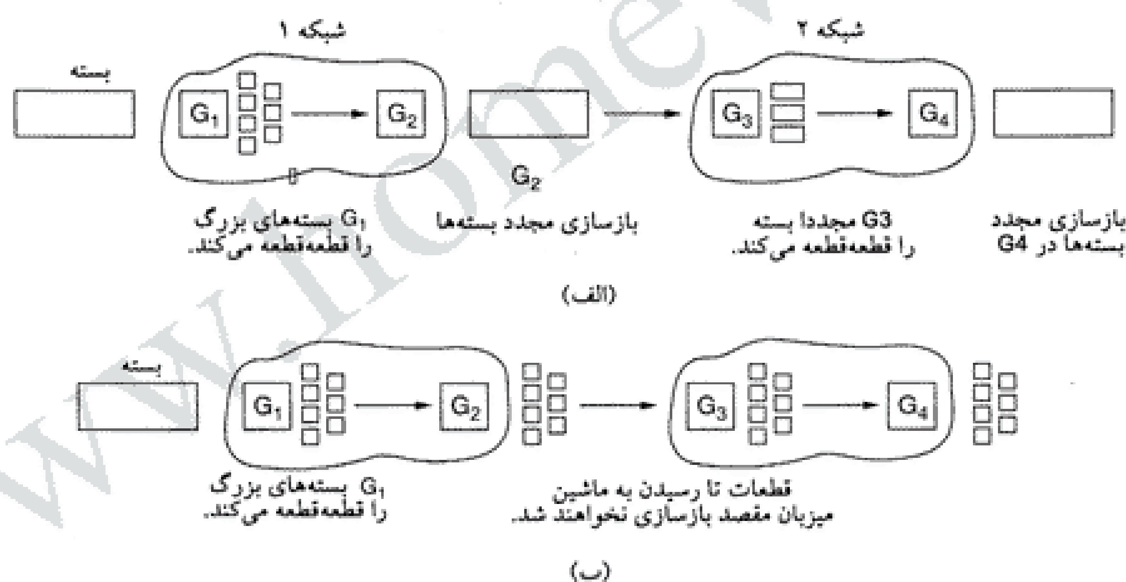
در هر شبکه اندازهٔ هر بسته نمی‌تواند از یک حد مُجاز بیشتر باشد. این محدودیت، علل مختلفی دارد که از این میان می‌توان به موارد ذیل اشاره کرد:

۱. سخت‌افزار (مثلاً طول فریم اترنت نمی‌تواند از حدود ۱۵۰۰ بایت بیشتر باشد.)
۲. سیستم عامل (مثلاً بافرها ۵۱۲ بایتی هستند.)
۳. پروتکل (مثلاً تعداد بیتهای فیلدی که میزان داده‌های موجود در هر بسته را مشخص می‌کند کم است.)
۴. سازگاری با برخی از استانداردهای بین‌المللی
۵. تمایل به کاهش حجم ارسال مجدد داده‌هایی که در اثر خطای انتقال از بین می‌روند
۶. اجتناب از اشغال بیش از حد کانال توسط یک بستهٔ بزرگ

نتیجهٔ این عوامل آن می‌شود که دست طراحان شبکه در انتخاب حداکثر طول بسته‌ها باز نیست. حداکثر طول داده‌هایی که می‌تواند درون یک بسته قرار بگیرد از ۴۸ بایت (در سلولهای ATM) تا ۶۵۵۱۵ بایت (در بسته‌های IP) متغیر است اگرچه طول داده‌ها در لایه‌های بالاتر، می‌تواند از این هم بیشتر باشد.

یکی از مشکلات بدیهی، زمانی رخ می‌دهد که یک بستهٔ بزرگ بخواهد به شبکه‌ای وارد شود که طول حداکثر بستهٔ آن کوچک است. یک راه حل آن است که از همان ابتدا این اطمینان حاصل شود که چنین مشکلی پیش نمی‌آید. به عبارت دیگر در اتصال شبکه‌ها، از یک الگوریتم مسیریابی استفاده شود که از ارسال بسته‌ها به شبکه‌ای که از عهدهٔ پذیرش آن بر نمی‌آید اجتناب کند، ولیکن این راه حل همیشه عملی نیست: اگر همهٔ بسته‌های تولید شده توسط مبداء، دارای اندازه‌ای باشند که شبکهٔ مقصد از عهدهٔ دریافت آن بر نمی‌آید چه اتفاقی می‌افتد؟ الگوریتم مسیریابی به ندرت می‌تواند چنین مقصدی را نادیده گرفته و آنرا کنار بگذارد.

اصولاً در چنین مواقعی می‌توان برای حل مشکل، به «دروازه» (Gateway) اجازه داد که بسته‌ها را به چند «قطعه» (Fragment) شکسته و هر یک از این قطعات را در پوشش یک بستهٔ مستقل ارسال کند؛ ولیکن والدین کودکان خردسال به خوبی می‌دانند که تکه‌تکه کردن اشیاء بزرگ به قطعات کوچک ساده‌تر از سرهم آنهاست!!! (فیزیکدانان برای این پدیده نامی انتخاب کرده‌اند: قانون دوم ترمودینامیک) شبکه‌های سوئیچ بسته نیز برای بازسازی قطعات به بستهٔ اصلی با مشکل روبرو هستند.



شکل ۵-۵۰. (الف) قطعه‌قطعه کردن نامرئی (شفاف) (ب) قطعه‌قطعه کردن غیرشفاف (مرئی).

برای بازسازی قطعات و تشکیل بستهٔ اصلی، دو استراتژی متضاد قابل اعمال است: استراتژی اول آن است که قطعه‌قطعه‌سازی بسته‌ها در شبکه‌ای که بسته‌های کوچک دارد به گونه انجام شود که این عمل از دید مقصد نهایی بسته، پنهان بماند. در این راهکار که در شکل ۵-۵۰-الف نشان داده شده است، در هر شبکه با بستهٔ کوچک یک «دروازه» (یا به عبارتی یک مسیریاب خاص) وجود دارد که واسط دیگر شبکه‌هاست. وقتی بسته‌ای با طول بیش از حدّ به این دروازه می‌رسد، آن را قطعه‌قطعه می‌کند؛ سپس هر یک از قطعات به آدرس دروازهٔ خروجی در مقصد ارسال شده و در آنجا قطعات به شکل اصلی بازسازی می‌شوند. بدین ترتیب عبور بسته‌های بزرگ از شبکه‌ای با بستهٔ کوچک، از دید شبکه‌های بیرونی پنهان می‌ماند؛ بعبارتی شبکه‌های بیرونی از قطعه‌قطعه شدن بسته‌ها آگاه نمی‌شوند. به عنوان مثال، شبکهٔ ATM دارای سخت‌افزار خاصی است که به صورت نامرئی، بسته‌ها را به سلولهای

۵۳ بایتی شکسته و در طرف مقابل آنها را به بستهٔ اصلی بازسازی می‌نماید. در شبکهٔ ATM به عمل شکستن بسته‌ها اصطلاحاً «**قطعه‌بندی**» (Segmentation) گفته می‌شود و مفهومی معادل با آنچه که در بالا بدان اشاره شد دارد، بلکه فقط جزئیات پیاده‌سازی آن متفاوت است.

استراتژی قطعه‌قطعه کردن بسته به صورت نامرئی، اگرچه ساده و سرراست است ولیکن مشکلاتی را در بر دارد. اولین مورد آن است که دروازهٔ خروجی باید تشخیص بدهد که آیا تمام قطعات یک بسته را دریافت کرده است یا آنکه هنوز قطعاتی در راه هستند و به همین دلیل باید یک بیت خاص پایان دنبالهٔ قطعات هر بسته را مشخص نماید. مورد دیگر *آن است که تمام قطعات یک بسته بایستی از یک دروازهٔ مشابه و یکسان خارج شوند.* وقتی اجازه نمی‌دهیم که قطعات یک بسته برای رسیدن به مقصد نهایی خود از مسیرهای مختلفی حرکت کنند، بخشی از کارآیی و بهینگی را از دست خواهیم داد. آخرین اشکال آنست که وقتی یک بستهٔ بزرگ مجبور است مکرراً از شبکه‌هایی بگذرد که آن را قطعه‌قطعه و بازسازی می‌کنند، سربار نسبتاً زیادی تحمیل خواهد شد. به هر تقدیر شبکه‌ای مثل ATM به قطعه‌قطعه‌سازی نامرئی [استراتژی فوق‌الذکر] نیاز دارد.

استراتژی دیگر برای قطعه‌قطعه کردن بسته‌ها آنست که از بازسازی بسته‌ها در دروازه‌های میانی اجتناب شود: هر گاه بسته‌ای قطعه قطعه شد، با هر یک از قطعات به مثابهٔ یک بستهٔ واقعی و اصلی رفتار شود. در این استراتژی به نحوی که در شکل ۵-۵۰-ب نشان داده شده تمام قطعات به همان نحو از دروازهٔ خروجی شبکه گذر می‌کنند. بازسازی بسته‌ها فقط در ماشین مقصد انجام می‌شود. IP بهمین نحو عمل می‌کند.[۱]

قطعه‌قطعه‌سازی غیرشفاف نیز با اشکالاتی مواجه است. به عنوان مثال، هر ماشین باید قادر به بازسازی بسته باشد. مشکل دیگر آنست که وقتی یک بستهٔ بزرگ به قطعات کوچک تقسیم می‌شود، سربار داده‌ها افزایش می‌یابد زیرا هر قطعه نیاز به سرآیند مستقل دارد، در حالی که در استراتژی اول به محض خروج قطعات از شبکه‌ای با بستهٔ کوچک، سربار تحمیل شده حذف می‌شود ولیکن در این روش سربار ناشی از سرآیند اضافی، تا خاتمهٔ مسیر باقی خواهد ماند. مزیت روش قطعه‌قطعه‌سازی غیرشفاف آن است که قطعات هر بسته می‌توانند از مسیری مجزا و دروازه‌های خروجی متفاوت به سوی مقصد نهایی خود هدایت شوند و طبعاً کارآیی بالاتر و بهینه‌تری حاصل خواهد شد. البته اگر شبکه‌های مختلف طبق مدل مدار مجازی به یکدیگر متصل و ملحق شده باشند [بخش ۵-۵-۳] این مزیت بکار نخواهد آمد.

هرگاه بسته‌ای قطعه‌قطعه شود، قطعات آن باید به نحوی شماره‌گذاری شوند که بتوان ترتیب اصلی داده‌ها را بازیابی کرد. یکی از روشهای شماره‌گذاری، روش درختی است: اگر بستهٔ شمارهٔ 0 نیاز به قطعه شدن داشته باشد قطعات به صورت 0.2 , 0.1 , 0.0 و به همین ترتیب، شماره‌گذاری می‌شوند. اگر هر یک از این قطعات، باز هم نیاز به شکسته شدن داشته باشند ترتیب شماره‌ها (از چپ به راست) به صورت ... ,0.1.3 , 0.1.2 , 0.1.1 , 0.1.0 , ... , 0.0.2 , 0.0.1 , 0.0.0 خواهد بود. اگر برای این منظور به تعداد کافی فیلد در سرآیند بسته پیش‌بینی شده باشد، الگوی فوق این اطمینان را می‌دهد که بتوان بسته‌ها را در مقصد بازسازی کرد. (حتی اگر قطعات به ترتیب دریافت نشوند.)

ولی اگر در یکی از شبکه‌های بین راه قطعه‌ای گم شود یا حذف گردد نیاز است که مجدداً کل بسته از مبداء آن ارسال و از نو قطعه قطعه و ارسال شود. فرض کنید که یک بستهٔ ۱۰۲۴ بیتی در ابتدا به چهار قطعهٔ مساوی با شماره‌های 0.0 ، 0.1 ، 0.2 و 0.3 تقسیم شده باشد. اگر قطعهٔ 0.1 از دست برود ولی بقیه قطعات سالم به مقصد

---

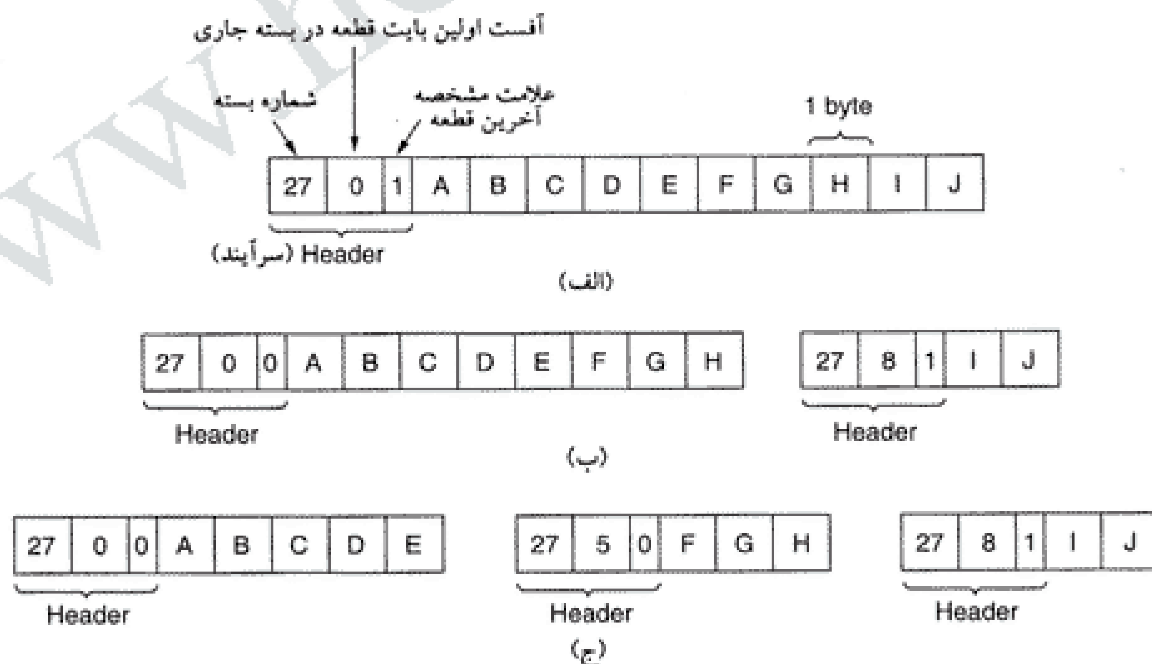
۱. بدین استراتژی، قطعه قطعه سازی غیرشفاف و مرئی (Nontransparent Fargmentation) گفته می‌شود. هرگاه بسته‌ای در هر نقطه از مسیر قطعه‌قطعه شد هیچکسی آنها را بازسازی نخواهد کرد مگر ماشین مقصدنهایی. -م

برسند، پس از مدتی مهلت فرستندهٔ بسته به سر آمده و کل بسته را از نو ارسال می‌کند. حال فرض کنید که به ناگاه محدودیت طول بسته از ۲۵۶ بیت به ۵۱۲ بیت افزایش پیدا کرده و در اینجا بسته به جای آنکه ۴ قسمت شود به دو قطعه تقسیم گردد. حال وقتی بستهٔ شمارهٔ 0.1 به مقصد می‌رسد گیرنده تصور می‌کند که تمام چهار قطعه‌ای که منتظر آن بوده، تکمیل شده و بسته را به صورت اشتباه بازسازی می‌کند!

یک روش شماره‌گذاری متفاوت و بهتر آن است که در پروتکل شبکه، یک اندازهٔ پایه و حداقل برای بسته‌ها تعریف شود و این اندازه به قدری کوچک باشد که بسته بتواند از هر شبکه‌ای عبور نماید. هرگاه بسته‌ای قطعه‌قطعه می‌شود، اندازهٔ تمام قطعات به مقدار پایه تنظیم خواهد شد (به استثنای قطعهٔ آخر که ممکن است کوچکتر باشد). در سرآیند هر یک از قطعات باید شمارهٔ بستهٔ اصلی و همچنین شمارهٔ قطعه مشخص شده باشد. همچنین باید در سرآیند قطعه یک بیت در نظر گرفته شود تا بتوان آخرین قطعهٔ هر بسته را مشخص نمود. (یعنی هر بسته را فقط یکبار می‌توان قطعه قطعه کرد.) در این روش، در سرآیند هر بسته به دو فیلد شماره ترتیب نیاز است: یکی برای شمارهٔ بستهٔ اصلی و دیگری برای شمارهٔ قطعه.

می‌توان یک حالت بینابین برای شماره‌گذاری بسته‌ها انتخاب کرد: به جای آن که برای هر قطعه شمارهٔ آن در سرآیند هر قطعه درج شود، آفست آن قطعه در بستهٔ اصلی، مشخص شود. [یعنی آن که مشخص شود قطعهٔ جاری در کجای بستهٔ اصلی قرار می‌گیرد.] اگر مطابق شکل ۵-۵۱ بجای شمارهٔ هر قطعه، آفست محل قرار گرفتن قطعه در بستهٔ اصلی در نظر گرفته شود می‌توان بسته را به هر اندازهٔ ممکن (حتی یک بایت) کوچک کرد.

در برخی از پروتکلهای شبکه بطور عام از این روش استفاده شده است حتی تا جایی که کل داده‌های ارسالی بر روی یک مدار مجازی به عنوان یک بستهٔ بسیار بزرگ تلقی می‌شود و شمارهٔ هر قطعه، شمارهٔ ترتیب «اولین بایت قطعه» نسبت به «اولین بایت بسته» فرض می‌شود. [یعنی محل قرار گرفتن قطعه نسبت به شماره اولین بایت بستهٔ اصلی مشخص می‌شود.]



شکل ۵-۵۱. قطعه‌قطعه‌سازی بسته‌ها با فرض آنکه مبنای پایهٔ طول داده‌ها ۱ بایت باشد. (الف) بستهٔ اصلی حاوی ده بایت داده. (ب) مجموعهٔ قطعات پس از عبور از شبکه‌ای که در آن طول حداکثر بسته‌ها با احتساب سرآیند، ۸ بایت است. (ج) مجموعهٔ قطعات پس از عبور از «دروازه‌ای» (Gateway) که در آن طول حداکثر بسته‌ها، ۵ بایت است.

۶ـ۵ لایهٔ شبکه در اینترنت

قبل از آنکه به توصیف لایهٔ شبکه در اینترنت بپردازیم، مروری بر اصول طراحی آن در گذشته و دلایل موفقیت آن در حال حاضر، خالی از لطف نیست. اصولی که به نظر می‌رسد اکنون به دست فراموشی سپرده شده‌اند. این اصول و قواعد در سند RFC 1958 تشریح شده‌اند و مطالعه آن بسیار ارزشمند است. (مطالعهٔ آن برای طراحان پروتکل باید الزامی شود و در پایان نیز از آنان براساس همین سند امتحان به عمل آید!) این RFC بشدت تحت تأثیر افکاری است که دو محقق Clark (۱۹۸۸) و Saltzer (همکاران و ۱۹۸۴) ارائه نموده‌اند. در اینجا به اختصار ده مورد از اصول اساسی طراحی پروتکل لایهٔ شبکه را (به ترتیب اهمیت) بررسی می‌نماییم:

۱. **اطمینان از عملکرد صحیح:** طراحی یا استاندارد را هیچگاه نهایی و مختومه فرض نکنید مگر آنکه چندین نسخهٔ اولیه و آزمایشی (پروتوتایپ) آن بتوانند با یکدیگر مبادلهٔ داده نمایند. بسیار اتفاق افتاده که طراحان شبکه یک استاندارد ۱۰۰۰ صفحه‌ای تدوین و آنرا به تایید رسانده‌اند ولی بعداً متوجه اشکالات اساسی و عدم عملکرد آن شده و مجبور به نوشتن نسخهٔ ۱.۱ استاندارد خود شده‌اند. این روش به نتیجه نخواهد رسید.

۲. **پروتکل را ساده طراحی کنید:** در هر کجا که تردید دارید، از ساده‌ترین راه حل بهره بگیرید. «ویلیام اُوکَم» این اصل را در قرن چهاردهم بیان کرده است. در یک عبارت امروزی و مدرن: «حداقل ویژگیهای زائد». اگر داشتن یک ویژگی، حیاتی و اساسی نیست آنرا رها کنید بالاخص زمانی که این ویژگی را می‌توان براساس ترکیبی از ویژگیهای دیگر بدست آورد.

۳. **تصمیمات روشن و شفاف بگیرید:** اگر برای انجام یک کار چندین راه حل پیش رو دارید فقط یکی را انتخاب نمایید. اگر دو یا چند روش را مدّنظر قرار بدهید برای خود مشکل تراشیده‌اید. امروزه بسیاری از استانداردها دارای گزینه‌های متعدد، حالات و پارامترهای مختلف هستند چرا که هر یک از طراحان، بر آنکه روش آنها بهترین است، پافشاری کرده‌اند. طراحان باید با قدرت در برابر این گرایشات فردی مقاومت کرده و فقط بگویند نه!!

۴. **طراحی شما ماجولار باشد:** این اصل بدین نظریه منتهی خواهد شد که باید پشته‌ای از پروتکلها را داشت و هر لایه مستقل از دیگر لایه‌ها باشد. بدین ترتیب هر گاه نیاز شد یک ماجول یا یک لایه تغییر کند، بقیهٔ لایه‌ها تحت تأثیر قرار نخواهند گرفت.

۵. **ناهمگونی عناصر را مدّنظر قرار بدهید:** در یک شبکهٔ بزرگ، انواع سخت‌افزار، امکانات مخابراتی و برنامه‌های کاربردی پیدا می‌شود. برای آن که بتوان همهٔ آنها را به خدمت گرفت و اداره کرد، طراحی شبکه باید ساده، عمومی و قابل انعطاف باشد.

۶. **از گزینه‌ها و پارامترهای ثابت پرهیز نمایید:** اگر در جایی الزاماً به تعریف پارامتر نیاز دارید (مثلاً تعریف طول حداکثر بسته‌ها) بهترین کار آن است که به جای تعریف پارامترهای ثابت اجازه بدهید فرستنده و گیرنده با یکدیگر مذاکره و بر سر مقدار آن توافق کنند.

۷. **به دنبال طراحی خوب باشید:** لازم نیست که یک طراحی خوب ایده‌آل و بی‌نقص باشد: بسیاری از طراحان، طرح خوبی را در اختیار دارند ولی طرح آنان نمی‌تواند از عهدهٔ برخی از موارد نادر و عجیب برآید. به جای آن که اینگونه طرحها را بهم بریزید بایستی در پی یک طراحی خوب بوده و از خود در مقابل خواسته‌های عجیب و غریب سلب مسئولیت نمایید.

۸. **هنگام «ارسال» سخت گیر و دقیق و هنگام «دریافت» با تحمل باشید:** به عبارتی دیگر فقط بسته‌هایی را

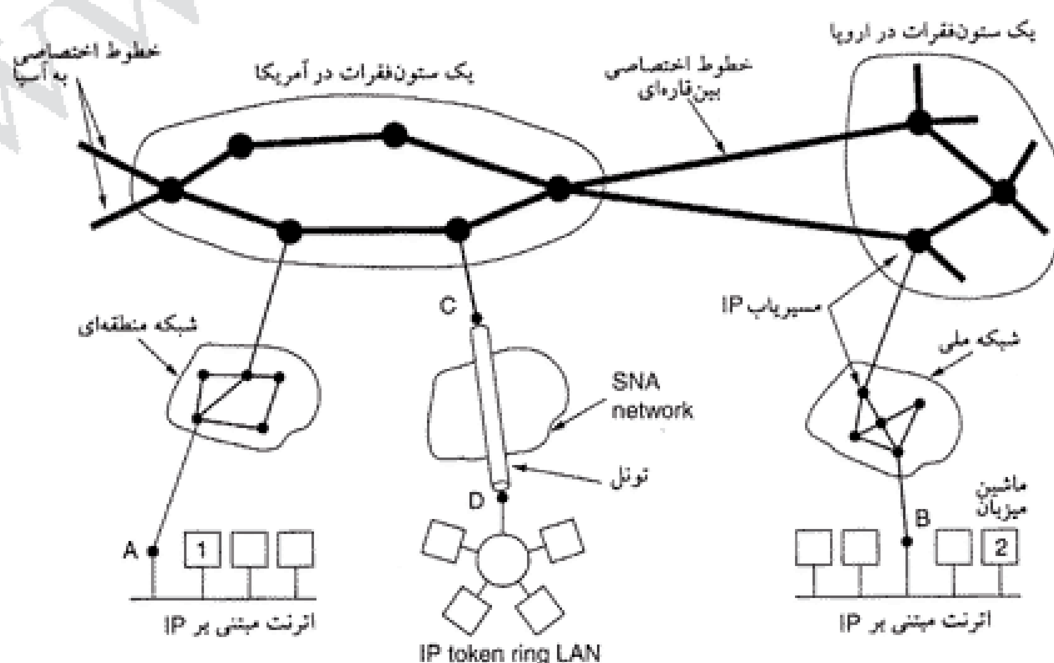
بر روی شبکه بفرستید که دقیقاً منطبق با استاندارد است ولی انتظار آن را داشته باشید که بسته‌های دریافتی انطباق کامل با استاندارد نداشته باشد.

۹. **در اندیشهٔ قابلیت گسترش و توسعه‌پذیری باشید:** اگر یک سیستم مجبور به مدیریت میلیونها ماشین و میلیاردها کاربر است، استفاده از هیچ نوع پایگاه اطلاعات مرکزی قابل تحمل و به صلاح نیست و باید بار آن تا حد ممکن بر روی منابع موجود پخش شود.

۱۰. **به کارآیی و هزینه دقت داشته باشید:** اگر شبکه‌ای کارآیی پایین یا هزینهٔ سرسام‌آور داشته باشد هیچکس از آن استقبال نخواهد کرد.

حال اصول کلی طراحی را کنار گذاشته و به جزئیات لایهٔ شبکه در اینترنت خواهیم پرداخت. از دیدگاه لایهٔ شبکه، اینترنت را می‌توان مجموعه‌ای از زیرشبکه‌ها یا اصطلاحاً «شبکه‌های خودمختار» در نظر گرفت که بهم متصل شده‌اند. هیچ ساختار مشخص و قطعی بر اینترنت حاکم نیست ولی چندین ستون‌فقرات (Backbone) برای آن وجود دارد. ستون‌فقرات از خطوط با پهنای باند بسیار بالا و مسیریابهای سریع تشکیل شده است. شبکه‌های منطقه‌ای (Regional Network) به ستون‌فقرات اینترنت متصل می‌شوند و شبکه‌های محلی دانشگاهها، شرکتها و مؤسسات ارائه دهندهٔ خدمات اینترنت (ISP) با شبکه‌های منطقه‌ای در ارتباط هستند. نمادی از این ساختار نسبتاً سلسله‌مراتبی در شکل ۵-۵۲ نشان داده شده است.

آنچه که تمام اجزاء شبکهٔ اینترنت را به هم چسبانده است همان پروتکل IP است. (IP/Internet Protocol) برخلاف بسیاری از پروتکلهای قدیمی لایهٔ شبکه، این پروتکل از صفر طراحی شده و از همان ابتدا در تفکر وصل شبکه‌ها به یکدیگر بوده است. برای آن که بتوانید وظیفهٔ لایهٔ شبکه را در ذهن خود مجسم کنید بدان بیندیشید که این لایه حداکثر تلاش خود را می‌کند تا یک دیتاگرام را از مبداء به مقصد برساند. [دیتاگرام را یک تودهٔ اطلاعات مستقل و دارای هویت در نظر بگیرید.] IP رسیدن بسته‌ها را تضمین نمی‌کند (بلکه حداکثر تلاش خود را مصروف آن می‌کند) و اینکه آیا ماشینهای مبداء و مقصد بر روی یک شبکه واقعند یا آن که شبکه‌های دیگری در این بین قرار گرفته‌اند، مهم نیست.

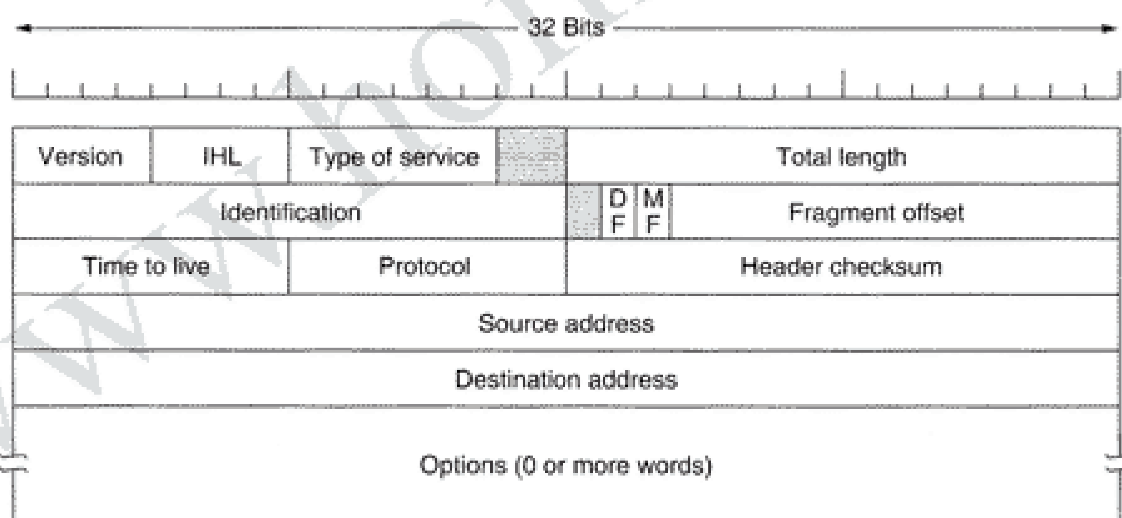


شکل ۵-۵۲. اینترنت مجموعهٔ تعداد بی‌شماری شبکه است.

ارتباط در اینترنت بدین ترتیب است که: لایهٔ انتقال یک دنباله از داده‌ها را تحویل گرفته و آنها را به چند دیتاگرام تقسیم می‌کند. در تئوری، طول حداکثر هر دیتاگرام می‌تواند تا ۶۴ کیلوبایت باشد ولی در عمل، بطور معمول حول و حوش ۱۵۰۰ بایت است (تا بتواند در یک فریم اترنت جا بگیرد). هر دیتاگرام از طریق اینترنت ارسال می‌شود و در صورت نیاز ، در طول مسیر به قطعات کوچکتر شکسته می‌شود. وقتی تمام قطعات به ماشین مقصد رسیدند توسط لایهٔ شبکه (IP) به دیتاگرام اصلی بازسازی می‌شود. این دیتاگرام تحویل لایهٔ انتقال شده و توسط این لایه به پروسهٔ گیرنده تسلیم می‌شود. همانگونه که در شکل ۵-۵۲ می‌بینید، بسته‌ای که از ماشین ۱ منشاء گرفته برای رسیدن به ماشین ۲ باید از شش شبکه بگذرد. در عمل این تعداد از شش تا هم بیشتر است.

۵-۶-۱ پروتکل IP

برای شروع به تحلیل پروتکل IP ، بهترین کار آن است که قالب بسته‌های IP را بررسی نماییم.[۱] یک بستهٔ IP در برگیرندهٔ دو بخش سرآیند و بخش محتوی است. سرآیند هر بسته دارای یک بخش ثابت ۲۰ بایتی و یک بخش اختیاری با طول متغیر است. ساختار سرآیند بستهٔ IP در شکل ۵-۵۳ نشان داده شده است. ترتیب ارسال بستهٔ IP از بایت پرارزش به کم‌ارزش است یعنی بایتها به ترتیب از چپ به راست ارسال می‌شوند؛ بنابراین اولین فیلدی که ارسال می‌شود فیلد Version (شمارهٔ نسخهٔ پروتکل) است. ماشینهای SPARC داده‌ها را از پرارزش به کم ارزش ذخیره و ارسال می‌کنند و به آنها ماشینهای Big Endian گفته می‌شود در حالیکه ماشینهایی مثل پنتیوم برعکس هستند یعنی داده‌ها را از کم‌ارزش به پرارزش ذخیره و ارسال می‌کنند (ماشینهای Little Endian) و بالطبع نرم‌افزار اینگونه ماشینها باید قبل از ارسال یا پس از دریافت، تبدیل لازم را انجام بدهند.



شکل ۵-۵۳. سرآیند IPv4 در پروتکل اینترنت.

فیلد Version مشخص می‌کند که بسته براساس چه نسخه‌ای از پروتکل IP سازماندهی و ارسال شده است. با قرار دادن شمارهٔ نسخهٔ پروتکل در هر دیتاگرام می‌توان فرآیند گذر از یک نسخهٔ قدیمی به نسخهٔ جدید را به صورت تدریجی و در خلال چندین سال به انجام رساند و در این مدت برخی از ماشینها از نسخهٔ قدیمی و برخی از نسخهٔ جدید استفاده کنند. اکنون در حال گذر از IPv4 به IPv6 هستیم و اگرچه سالهاست که انتظار این جایگزینی می‌رود ولی به نظر می‌رسد هنوز هم به سالها وقت نیاز دارد.

۱. به بسته‌های IP اصطلاحاً «دیتا گرام IP» گفته می‌شود. «دیتا گرام IP» یک قطعه دادهٔ دارای هویت و شناسنامه است. -م

(Durand, 2001; Wiljakka, 2002; Waddington and Chang, 2002) حتی برخی افراد معتقدند این تغییر هیچگاه محقّق نخواهد شد. (Weiser , 2001) مضاف بر این، IPv5 نیز یک بعنوان پروتکل آزمایشی برای «انتقال بی‌درنگ استریم داده»[1] طراحی گردید ولیکن استفادهٔ چندانی از آن نشد.

از آنجایی که طول سرآیند ثابت نیست لذا فیلد IHL بدان منظور در نظر گرفته شده که مشخص کند طول سرآیند (بر مبنای کلمات ۳۲ بیتی) چقدر است. مقدار حداقل این فیلد (یعنی زمانی که هیچ داده‌ای در بخش اختیاری سرآیند وجود ندارد) معادل ۵ است. حداکثر مقدار این فیلد ۴ بیتی، ۱۵ است و طبعاً طول سرآیند به ۶۰ بایت (۴×۱۵) محدود می‌شود؛ یعنی حداکثر طول بخش اختیاری ۴۰ بایت (۲۰_۶۰) است. این فضای ۴۰ بایتی برای برخی از گزینه‌هائی که می‌توان درون این فیلد اختیاری درج کرد (مثل گزینهٔ ثبت مسیر طی شده توسط بسته) بسیار ناکافی است و عملاً فیلد اختیاری را بی‌استفاده کرده است.

فیلد Type of Service (نوع خدمات)، یکی از معدود فیلدهایی است که مضمون و عملکرد آن در طول این سالها تغییر کرده است. از ابتدا (و همچنین اکنون) این فیلد به منظور مشخص کردن کلاسهای مختلف خدمات مدّنظر بوده است. در این فیلد می‌توان ترکیبی از خدمات مثل قابلیت اعتماد (Reliability) و سرعت (Speed) را برای بسته تقاضا کرد. برای انتقال دیجیتال صدا، تحویل سریع داده‌ها ارجح‌تر از تحویل مطمئن و بدون خطاست. برعکس، برای انتقال فایل ارسال بدون خطا بسیار مهمتر از سرعت انتقال است.

فیلد شش بیتی Type of Service (به ترتیب از چپ به راست) در برگیرندهٔ یک فیلد سه بیتی به نام Precedence (فیلد تقدم) و سه بیت علامت به نامهای D و T و R است. فیلد Precedence اولویت بسته را مشخص می‌کند: از صفر (اولویت معمولی) تا ۷ (بالاترین اولویت برای بسته‌های کنترلی شبکه). سه بیت علامت به ماشین میزبان اجازه می‌دهد که از بین سه ویژگی (D: تأخیر، T: ظرفیت خروجی، R: قابلیت اعتماد) نیازهای خود را توصیف نماید. از دیدگاه تئوری این فیلدها امکان آنرا فراهم آورده‌اند که مسیریاب مثلاً از بین یک خط ماهواره‌ای با ظرفیت خروجی بالا و تأخیر زیاد و همچنین یک خط اجاره‌ای با ظرفیت خروجی کم و تأخیر پایین، انتخاب مناسبی داشته باشد، ولیکن در عمل مسیریابها فیلد Type of service را نادیده می‌گیرند.

عاقبت IETF متقاعد شد که مضمون این فیلد را تغییر بدهد تا براساس آن بتوان رده‌های مختلف خدمات مورد نیاز را توصیف کرد. ۶ بیت این فیلد برای مشخص کردن ردهٔ کلاس خدماتی است که بسته بدان کلاس تعلق دارد. این کلاسها که قبلاً در بخش ۵-۴-۴ معرفی شدند، شامل: ۴ اولویت صف‌بندی، ۳ احتمال حذف بسته و تعدادی کلاس برای سازگاری با قبل هستند.

فیلد Total Length طول کل بسته را (شامل سرآیند و داده) مشخص می‌نماید. حداکثر طول یک بسته ۶۵۵۳۵ بایت است. در حال حاضر این مقدار حداکثر کفایت می‌کند ولی در آینده با رواج اترنت گیگابیت ممکن است به دیتاگرام با طول بیشتری نیاز باشد.

فیلد Identification بدان جهت نیاز است که مشخص کنیم قطعهٔ دریافتی، به کدام دیتاگرام متعلق است. تمام قطعات یک دیتاگرام واحد دارای مقداری مشابه در فیلد Identification هستند. [بخش ۵-۵-۷ را مطالعه نمایید.] در ادامه یک بیت بلااستفاده و سپس یک فیلد تک بیتی به نام DF[2] قرار گرفته است. این بیت به مسیریابها فرمان می‌دهد که دیتاگرام (بسته) جاری را قطعه‌قطعه نکنند چرا که مقصد از بازسازی آن عاجز است. مثلاً وقتی

---

۱. Real-Time Stream Protocol

۲. DF مخفف کلمات Don't Fragment به معنای عدم قطعه‌قطعه‌سازی بسته است.

کامپیوتری بوت می‌شود، ROM بوت‌کنندهٔ آن ممکن است تقاضا کند که «تصویری از حافظه» (Memory Image) برای او ارسال شود.[۱] با علامتگذاری در بیت DF ، فرستنده مطمئن خواهد شد که کل بسته به صورت یکجا تحویل خواهد شد، حتی اگر این بسته به دلیل محدودیتهایی که مسیر بهینه (در هدایت بسته‌های بزرگ) دارد مجبور به عبور از مسیرهای غیربهینه شود. تمام ماشینها ملزم به پذیرش قطعات با طول ۵۷۶ بایت (یا کمتر) هستند. بیت MF (مخفف More Fragment) به معنای آنست که دنبالهٔ قطعات هنوز ادامه دارد. در تمام قطعات به استثنای قطعهٔ آخر این بیت ۱ است. به این بیت از آن جهت نیاز است که متوجه شویم آیا تمام قطعات یک دیتاگرام دریافت شده یا هنوز دنبالهٔ قطعات ادامه دارد.

فیلد Fragment Offset مشخص کنندهٔ آنست که قطعهٔ جاری در کجای دیتاگرام اصلی واقع شده است. طول تمام قطعات (به استثنای قطعهٔ آخر) باید ضریبی از ۸ بایت باشد. از آنجایی که این فیلد ۱۳ بیتی است حداکثر می‌توان ۸۱۹۲ قطعه در هر دیتاگرام داشت و بدین ترتیب طول حداکثر هر دیتاگرام ۶۵۵۳۶ بایت خواهد شد.

فیلد Time to live (زمان حیات بسته) شمارنده‌ای است که طول عمر بسته را تعیین می‌نماید. فرض بر آنست که این فیلد زمان را برحسب ثانیه مشخص کند و طول عمر بسته حداکثر ۲۵۵ ثانیه باشد. به ازای هر گام (یعنی عبور بسته از یک مسیریاب)، یک واحد از این فیلد کاسته می‌شود. همچنین فرض شده که هر گاه بسته درون یک مسیریاب، زمان زیادی را در صف منتظر ماند تعداد بیشتری از این فیلد کم شود. با تمام این تفاصیل، در عمل فقط تعداد گام محاسبه می‌شود.[۲] به محض آن که مقدار این فیلد به صفر برسد بسته حذف شده و یک پیغام هشدار به مبداء آن برگردانده می‌شود. این ویژگی از سرگردان ماندن بسته‌ها در زیرشبکه (که در اثر اشتباه در جدول مسیریابی بروز می‌کند) جلوگیری می‌نماید.

هر گاه لایهٔ شبکه یک دیتاگرام کامل را بازسازی کرد باید بداند که چه کاری با آن بکند. فیلد Protocol (شمارهٔ پروتکل)، مشخص می‌کند که بسته را باید به کدام پروسه در لایهٔ انتقال تحویل داد. پروسه TCP یکی از گزینه‌هاست؛ ولی پروسهٔ UDP و چندین پروسهٔ دیگر نیز می‌توانند دیتاگرام را تحویل بگیرند. شمارهٔ پروتکلها جهانی و در سرتاسر اینترنت استاندارد هستند. پروتکلها و دیگر شماره‌های استاندارد در سند RFC 1700 ثبت شده است؛ این شماره‌ها را می‌توانید در پایگاه اطلاعاتی www.iana.org بدست بیاورید.

فیلد Header Checksum یک کُد کشف خطا برای سرآیند است. این کد کشف خطا که صرفاً سرآیند بسته را در برمی‌گیرد قادر است خطاهایی را که از خرابی در حافظهٔ مسیریاب ناشی می‌شود آشکار نماید. الگوریتم کشف خطا بدین ترتیب است که کل سرآیند، در همان بدو ورود در قالب نیم‌کلمات ۱۶ بیتی با یکدیگر جمع شده و نهایتاً «مکمل یک»[۳] آن محاسبه می‌شود. اگر سرآیند بسته بدون خطا باشد جمع تمام نیم کلمات ۱۶ بیتی سرآیند، باید صفر باشد. این الگوریتم قدرتمندتر از جمع معمولی است. دقت کنید که فیلد Header Checksum در هر گام باید از نو محاسبه شود چرا که حداقل یکی از فیلدهای سرآیند تغییر می‌کند (یعنی فیلد Time To Live) ولی برای سرعت بخشیدن به این محاسبه می‌توان از راهی میانبُر استفاده کرد.

فیلدهای Source Address (آدرس مبداء) و Destination Address (آدرس مقصد) شمارهٔ شبکه و شمارهٔ ماشین مبداء و مقصد را مشخص می‌کنند. این دو آدرس را در بخشی مجزا بررسی خواهیم کرد.

فیلد Options بدان منظور طراحی شده تا در نسخه‌های بعدی پروتکل بتوان اطلاعاتی را در سرآیند قرار داد که در نسخهٔ اصلی آن پیش‌بینی نشده است. همچنین می‌توان از آن برای آزمودن ایده‌های جدید بدون درهم

---

۱. تصویری از هستهٔ سیستم عامل که بصورت کدهای اجرایی ارسال می‌شود.

۲. یعنی امروزه فقط به ازای عبور بسته از یک مسیریاب، یک واحد از مقدار این فیلم کم می‌شود و زمان (برحسب ثانیه) هیچ تاثیری در این فیلد ندارد.

۳. One's Compelement

ریختن آن بهره گرفت. فیلد Option دارای طولی متغیر است؛ هر یک از گزینه‌های مندرج در این فیلد با یک کد یک بایتی شروع می‌شود تا ماهیت گزینه را مشخص کند. در برخی از گزینه‌ها پس از این کد یک بایتی، فیلد یک بایتی دیگر، طول گزینه را برحسب بایت مشخص می‌نماید. نهایتاً به فیلد Options تعدادی بایت اضافی و زائد افزوده می‌شود تا طول آن ضریبی از ۴ شود. در ابتدا فقط پنج گزینهٔ فهرست شده در شکل ۵-۵۴ تعریف شده بود ولی پس از آن تعدادی گزینهٔ دیگر بدان افزوده شد. فهرست کامل این گزینه‌ها در آدرس www.iana.org/assignments/ip-parameters در دسترس عموم قرار گرفته است.

گزینه	توصیف عملکرد
Security	میزان محرمانه بودن دیتاگرام جاری را تعیین می‌کند.
Strict source routing	فهرست کامل مسیری را که باید دنبال شود تعریف می‌کند.
Loose source routing	فهرستی از مسیریابها که بسته حتما باید از آنها رد شود.
Record route	مسیریابها را وادار می‌کند تا آدرس IP خود را در بسته درج کنند.
Timestamp	مسیریابها را وادار می‌کند تا آدرس IP خود را بهمراه مهر زمان در بسته درج کنند.

شکل ۵-۵۴. برخی از گزینه‌های IP.

گزینهٔ Security مشخص کنندهٔ آن است که اطلاعات موجود در بسته تا چه حد محرمانه است. از دیدگاه تئوری، کاربرد این گزینه آن است که مسیریابهای نظامی این بسته را از مسیرهایی که از کشورها یا مناطقی می‌گذرند که به زعم آنها نامطمئن و خطرناک هستند، هدایت نکنند. در عمل تمام مسیریابها این فیلد را نادیده می‌گیرند و تنها کاربرد عملی آن می‌تواند کمک به جاسوسان برای انتخاب بسته‌های مورد نظرشان باشد!

با گزینهٔ Strict Source Routing می‌توان مسیر کامل بین مبداء و مقصد را در قالب دنباله‌ای از آدرسهای IP مشخص کرد. بستهٔ ارسالی ملزم به عبور از این مسیر است. این گزینه به مدیران شبکه کمک می‌کند تا در هنگام خرابی جداول مسیریابی یا عملکرد غیرصحیح مسیریابها بتوانند بسته‌های اضطراری خود را به مقصد برسانند یا می‌تواند برای اندازه‌گیری زمانی و تخمین ترافیک یک مسیر مفید واقع شود.

با گزینهٔ Loose Source Routing، بسته ملزم به عبور از مسیریابهای مشخص (طبق ترتیب تعیین شده) می‌باشد ولی اجازه دارد در مسیر خود از مسیریابهای دیگری نیز که در فهرست نیامده، عبور کند. [یعنی در حقیقت فقط برخی از نقاط مسیر مشخص می‌شود و بقیهٔ نقاط توسط مسیریاب انتخاب می‌شود.] بطور معمول در این گزینه فقط تعدادی از مسیریابها مشخص می‌شوند و همین تعداد می‌تواند مسیر مورد نظر را تبیین نماید. به عنوان مثال برای آنکه بسته‌ای را مجبور کنیم برای رفتن از لندن به سیدنی به جای شرق از سمت غرب حرکت کند کافی است در این گزینه سه مسیریاب در نیویورک، لوس آنجلس و هانولولو مشخص شود. این گزینه زمانی سودمند خواهد بود که به دلائل سیاسی یا اقتصادی بخواهیم بسته‌ها از مسیرهایی که در برخی کشورهای خاصّ واقعند عبور نکنند.

گزینهٔ Record Route (ثبت مسیر) به مسیریابهای واقع بر روی مسیر تفهیم می‌کند که باید آدرس IP خود را در همین فیلد (فیلد Option) درج نمایند. این گزینه به مدیران شبکه کمک می‌کند تا بتوانند اشکالات احتمالی در الگوریتم مسیریابی را پیگیری و کشف نمایند (مثلاً: «چرا بسته‌ای که از هیوستون به دالاس روانه شده در توکیو دیده شده است؟») زمانی که برای اولین بار ARPANET راه‌اندازی شد کل مسیریابهای این شبکه ۹ تا بیشتر نبود و فضای چهل بایتی فیلد Options کفایت می‌کرد ولی امروزه به دلیل رشد سرسام‌آور شبکهٔ اینترنت این فضای چهل بایتی بسیار ناکافی و عملاً بی‌کاربرد است.

در آخر، گزینهٔ Timestamp (مُهر زمان) عملکردی شبیه به گزینهٔ قبلی (یعنی Record Route) دارد با این تفاوت که به غیر از ثبت آدرس IP هر مسیریاب، یک «مُهر زمان» ۳۲بیتی نیز در کنار آن درج می‌شود. این گزینه نیز برای اشکال‌زدایی از الگوریتمهای مسیریابی کاربرد دارد.

۵-۶-۲ آدرسهای IP

هر ماشین میزبان و هر مسیریاب در شبکهٔ اینترنت دارای یک آدرس IP است که در آن شمارهٔ شبکه و شمارهٔ ماشین، مشخص می‌شود. ترکیب این دو شماره، منحصر به فرد و یکتا است: بر اساس این اصل اساسی که هیچ دو ماشینی در اینترنت نباید آدرس IP یکسانی داشته باشند. تمام آدرسهای IP ، ۳۲بیتی هستند و برای مشخص کردن آدرس ماشین مبداء و مقصد، در درون فیلدهای مربوطه در بسته‌های IP قرار می‌گیرند. اشاره به این نکته بسیار مهم است که آدرس IP در حقیقت یک ماشین میزبان را مشخص نمی‌کند بلکه به یک «کارت واسط شبکه» اشاره دارد، فلذا اگر ماشینی بر روی دو شبکه واقع شده باشد باید دو آدرس IP داشته باشد ولی در عمل بیشتر ماشینها فقط بر روی یک شبکه قرار گرفته‌اند و طبعاً یک آدرس IP بیشتر ندارند.[1]

شکل ۵-۵۵. قالب آدرسهای IP.

در چند دههٔ ابتدایی، آدرسهای IP در پنج ردهٔ مختلف دسته‌بندی شده بودند. این پنج رده در شکل ۵-۵۵ نشان داده شده است. تخصیص فضای آدرس مطابق با این روش، اصطلاحاً روش «آدرس‌دهی دارای کلاس» (Classful Addressing) نامیده می‌شود. این روش امروزه کاربرد خود را از دست داده است [از حدود سال ۱۹۹۶] ولی هنوز هم در کتب مختلف به آن اشاره می‌شود. این روش آدرس‌دهی را به اختصار توضیح می‌دهیم.

کلاسهای A و B و C و D این امکان را فراهم آورده‌اند تا بتوان به ترتیب: ۱۲۸ شبکه با ۱۶ میلیون ماشین میزبان (در کلاس A)، ۱۶۳۸۴ شبکه با ۶۵۵۳۶ ماشین (در کلاس B) و حدود ۲ میلیون شبکه با ۲۵۶ ماشین (در کلاس C) را آدرس‌دهی کرد (اگرچه برخی از این آدرسها برای کارهای ویژه در نظر گرفته شده‌اند و به هیچ ماشینی منتسب نمی‌شوند.) همچنین در کلاس D از ارسال چندپخشی (Multicast) پشتیبانی می‌شود و می‌توان بسته‌ای را مستقیماً برای چندین ماشین در شبکه فرستاد.

---

۱. مسیریابها که چندین کارت واسط دارند بیش از یک آدرس IP خواهند داشت.

آدرسهایی که با چهار بیت ۱۱۱۱ شروع می‌شوند (یعنی کلاس E) هیچ استفاده‌ای ندارند و برای کاربردهای بعدی رزرو شده‌اند. امروزه بیش از ۵۰۰,۰۰۰ شبکه به اینترنت متصل شده‌اند و این تعداد سال به سال افزایش می‌یابد. شماره‌های شبکه [یعنی فیلد Network در آدرس IP] توسط یک موسسهٔ غیرانتفاعی به نام ICANN[1] مدیریت می‌شود تا تناقضی در انتساب شماره‌های تکراری پدید نیاید. ICANN تخصیص بخشهایی از فضای آدرس‌دهی را به تعدادی «مراکز مجاز منطقه‌ای» واگذار کرده تا آنها آدرسهای IP را بین سرویس دهنده‌های اینترنتی (ISP) و دیگر شرکتها توزیع نمایند.

آدرسهای IP که ۳۲ بیتی هستند معمولاً با نماد دهدهی نقطه‌دار نمایش داده می‌شوند. در این روش هر یک از چهار بایتِ آدرس به صورت مجزا و در مبنای ده (از صفر تا ۲۵۵) نوشته می‌شود. مثلاً آدرس ۳۲بیتی C0290614 (در مبنای شانزده) به صورت 192.41.6.20 نمایش داده می‌شود. کمترین آدرس IP معادل 0.0.0.0 و بزرگترین آن معادل با 255.255.255.255 است.

مقدار صفر و ۱- (یعنی فیلدی که تمام بیتهایش 1 است) معانی خاص دارد. شکل ۵-۵۶ آدرسهای خاص را مشخص کرده است. مقدار ۰ خود ماشین یا شبکه را مشخص می‌نماید.[2] مقدار ۱- برای پخش فراگیر (Broadcast) کاربرد دارد، یعنی اگر بیتهای آدرس مقصد بسته تماماً ۱ باشد، گیرندهٔ بسته، تمام ماشینهای آن شبکه می‌باشد.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		This host (همین ماشین)
0 0 . . . 0 0	Host	A host on this network (ماشینی بر روی همین شبکه)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		Broadcast on the local network (پخش فراگیر بر روی شبکه محلی)
Network	1 1 1 1 . . . 1 1 1 1	Broadcast on a distant network (پخش فراگیر برروی شبکه راه‌دور)
127	(Anything)	Loopback (آدرس حلقه بازگشت)

شکل ۵-۵۶. آدرسهای IP خاص.

آدرس IP معادل با 0.0.0.0 توسط ماشینهای میزبان و در حین راه‌اندازی (بوت) مورد استفاده قرار می‌گیرد. هر گاه شمارهٔ شبکه در آدرس IP صفر باشد چنین آدرسی به شبکهٔ فعلی اشاره می‌کند. این آدرسها اجازه می‌دهد که ماشینها بدون دانستن شمارهٔ شبکهٔ خود، به آن اشاره کنند [یعنی اگر ماشینی بخواهد برای ماشین دیگری در شبکهٔ خودش بسته‌ای بفرستد، دانستن شمارهٔ شبکه، مهم نیست]. البته باید هر ماشین، حداقل کلاس آدرس شبکهٔ خود را بداند تا تشخیص بدهد فیلد شمارهٔ شبکه چند بیتی است و آن را با صفر پر کند. آدرسی که تمام بیتهای آن ۱ است اجازه می‌دهد که بسته به صورت فراگیر بر روی شبکهٔ محلی پخش شود. [یعنی بسته‌ای با چنین آدرسی را همهٔ ماشینهای موجود بر روی شبکهٔ محلّی دریافت می‌دارند.] اگر شمارهٔ شبکه با یک مقدار درست تنظیم شده ولی تمام بیتهای شمارهٔ ماشین مساوی ۱ باشد می‌توان بسته‌ای را برای تمام ماشینهای یک شبکهٔ LAN، در هر کجای اینترنت ارسال نمود. (البته بسیاری از مسئولین شبکه این ویژگی را غیرفعال و ناممکن کرده‌اند.)

---

۱. Internet Corporation for Assigned Names and Numbers

۲. یعنی اگر فیلد Network در آدرس IP معادل صفر باشد به معنای خود شبکه و اگر فیلد Host صفر باشد به معنای خود آن ماشین است. بسته‌ای با این آدرسها از شبکه یا ماشین بیرون نخواهد رفت. -م

تمام آدرسهایی که به شکل 127.xx.yy.zz هستند به نام «**حلقهٔ بازگشت**» (Loopback) مشهورند و برای آزمایش نرم‌افزار، کنار گذاشته شده‌اند: هر بسته‌ای که به چنین آدرسهایی ارسال شود به صورت واقعی بر روی سیم منتقل نخواهد شد بلکه به صورت داخلی پردازش شده و با آن همانند بستهٔ ورودی رفتار می‌شود. این آدرس اجازه می‌دهد بدون آنکه ماشین فرستنده شمارهٔ خود را بداند برای خودش بسته بفرستد.

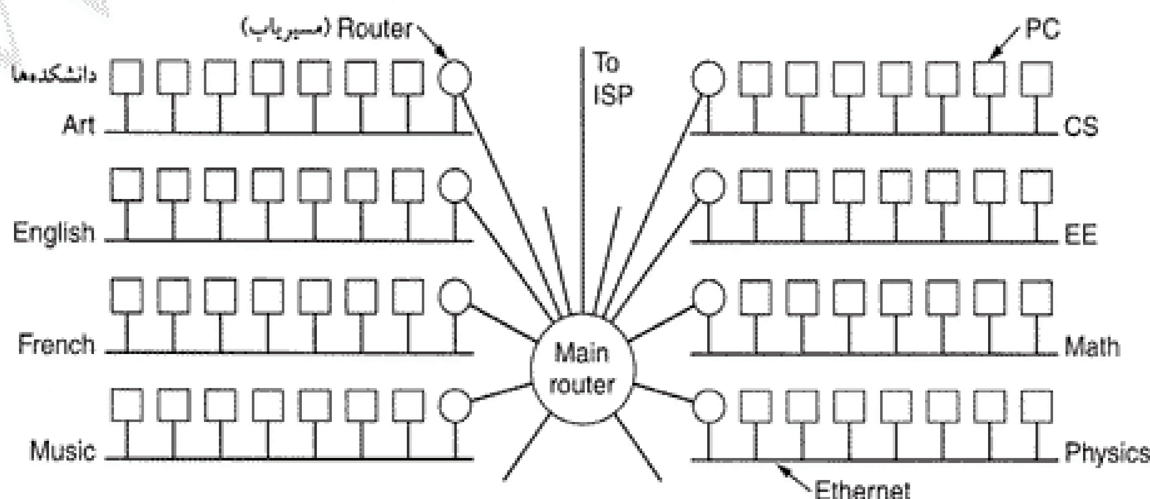
زیرشبکه‌ها

همانگونه که قبلاً بررسی کردیم تمام ماشینهای یک شبکه باید دارای شمارهٔ شبکهٔ یکسانی باشند. با رشد شبکه، این ویژگی در آدرس‌دهی IP، باعث بروز مشکلاتی می‌شود. به عنوان مثال دانشگاهی را مدّنظر قرار بدهید که در ابتدا با یک آدرس کلاس B شروع کرده و آن را در شبکهٔ اترنت متعلق به دانشکدهٔ علوم کامپیوتر بکار گرفته است. یک سال بعد دانشکدهٔ مهندسی برق می‌خواهد که به اینترنت متصل شود و با خرید یک تکرارکننده (Repeater) سعی می‌کند به شبکهٔ اترنت دانشکدهٔ علوم کامپیوتر متصل شود. با گذشت زمان، دانشکده‌های دیگر نیز تقاضای وصل به شبکه می‌کنند و بدین ترتیب به دلیل محدودیت در تعداد تکرارکننده‌های اترنت این کار غیرممکن می‌شود. بنابراین به سازماندهی متفاوتی نیاز است.

از طرف دیگر، ثبت و دریافت یک کلاس آدرس مجزا برای شبکه‌های جدید نیز دشوار است چراکه آدرسهای شبکه محدود و کم هستند و مضاف بر این، آدرس کلاس B اختصاص داده شده به این دانشگاه برای آدرس‌دهی به بیش از ۶۰۰۰۰ ماشین کافی است. مشکل اینجاست که یک کلاس A، B یا C به یک شبکهٔ واحد اشاره می‌کند نه به یک مجموعه از شبکه‌های LAN متصل بهم.

پس از آنکه سازمانهای متعددی به مشکل کمبود آدرس شبکه، دچار شدند یک تغییر بسیار کوچک در این آدرسها مشکل را حل کرد: راه حل آن است که اجازه بدهیم شبکه به چندین بخش مستقل داخلی تقسیم شده ولی کماکان در دنیای خارج به عنوان یک شبکهٔ واحد تلقی شود.

یک «شبکهٔ دانشگاهی» (Campus)، نمادی شبیه به شکل ۵-۵۷ دارد که در آن یک مسیریاب مرکزی از طرفی به یک ISP (یا یک شبکهٔ منطقه‌ای) و از طرف دیگر به چندین شبکه اترنت در دانشکده‌های مختلف وصل شده است. هر یک از شبکه‌های اترنت دارای یک مسیریاب محلی هستند که از طریق آن به مسیریاب مرکزی متصل شده‌اند. (البته ممکن است به جای مسیریاب مرکزی یک ستون فقرات قرار گرفته باشد ولیکن چگونگی اتصال مسیریابها در این مثال مهم نیست.)



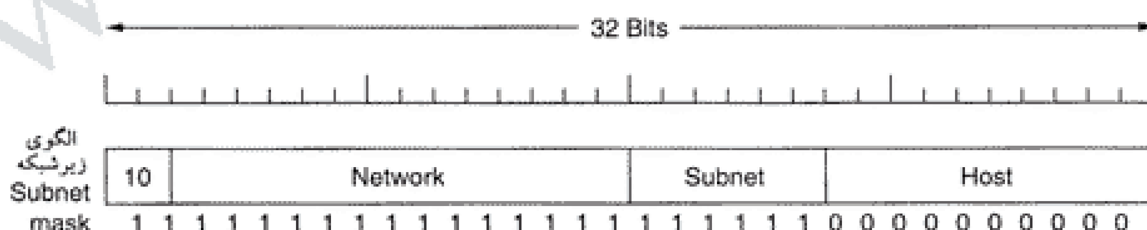
شکل ۵-۵۷. یک شبکهٔ دانشگاهی که شامل چندین LAN متعلق به دانشکده‌های مختلف است.

در ادبیات شبکهٔ اینترنت، به هر بخش از یک شبکهٔ بزرگ و خودمختار، اصطلاحاً «**زیرشبکه**» (Subnet) گفته می‌شود. (مثلاً در مثال بالا هر یک از شبکه‌های اترنت یک «زیرشبکه» تلقی می‌شوند.) همانطور که در فصل اول اشاره کردیم، این استعاره می‌تواند با واژهٔ «زیرشبکه» به معنای مجموعه‌ای از مسیریابها و خطوط ارتباطی در یک شبکه اشتباه شود. البته زمینه و مفاد یک بحث، معنای مورد نظر آن را مشخص خواهد کرد. در این بخش و بخش آتی، هر گاه واژهٔ «زیرشبکه» را بکار بردیم، تعریف جدید آن مدّنظر است.

وقتی بسته‌ای به مسیریاب مرکزی وارد می‌شود، مسیریاب چگونه تشخیص می‌دهد که آنرا به کدام زیرشبکه (اترنت) تحویل بدهد؟ یک راه آن است که جدولی با ۶۵۵۳۶ درایه (Entry) داشته باشیم و هر درایه محل هر ماشین میزبان را در کل شبکه مشخص کند. این ایده عملی است ولی به یک جدول بسیار بزرگ در مسیریاب مرکزی نیاز است و در صورت اضافه شدن، حذف یا تغییر محل ماشینها، باید تنظیمات لازم بصورت دستیْ انجام شود.

به جای این روش، الگوی جدیدی ابداع شده است. در این روش به جای آنکه یک کلاس B (با ۱۴ بیت برای شمارهٔ شبکه و ۱۶ بیت برای شمارهٔ ماشین) داشته باشیم، بخشی از فضای شانزده بیتی شمارهٔ ماشین برای تعیین «شمارهٔ زیرشبکه» در نظر گرفته می‌شود. به عنوان مثال اگر دانشگاه ۳۵ دانشکده داشته باشد، می‌تواند ۶ بیت را برای شمارهٔ زیرشبکه و ده بیت باقیمانده را برای شمارهٔ ماشین در نظر بگیرد. بدین ترتیب می‌تواند حداکثر ۶۴ زیرشبکه اترنت با حداکثر ۱۰۲۲ ماشین میزبان داشته باشد. (قبلاً اشاره شد که شماره‌های ۰ یا ۱- کاربرد خاص داشته و قابل استفاده نیست). این تقسیم بندی را بعداً می‌توان تغییر داد.

برای تعریف و پیاده‌سازی زیرشبکه‌ها، مسیریاب مرکزی به یک «**الگوی زیرشبکه**» (Subnet Mask) نیاز دارد تا مطابق با شکل ۵-۵۸بتوان مرز بین شمارهٔ ماشین و شمارهٔ شبکه و شمارهٔ زیرشبکه را مشخص و تفکیک کرد. الگوی زیرشبکه (Subnet Mask) نیز به صورت نماد نقطه‌دار دهدهی نوشته می‌شود. همچنین برای راحتی نمایش، می‌توان پس از آدرس IP یک علامت "/" گذاشت و پس از آن تعداد بیتهای ۱ موجود در الگوی زیرشبکه را تعیین کرد. به عنوان مثال در شکل ۵-۵۸ الگوی زیرشبکه را می‌توان به صورت 255.255.252.0 نوشت. راه دیگر آن است که از نماد 22/ استفاده نماییم که مشخص‌کننده آن است که الگوی زیرشبکه ۲۲ بیت طول دارد. (۲۲ بیت ۱ از سمت چپ به راست)



شکل ۵-۵۸. یک کلاس B به ۶۴ زیرشبکه بخش‌بندی شده است.

خارج از محدودهٔ این شبکه، این چنین تقسیم بندی و تعداد زیرشبکه‌ها مشهود و مشخص نیست لذا تعریف زیرشبکه جدید مستلزم آن نیست که با ICANN تماس گرفته شده یا تغییری در پایگاه اطلاعات خارج از آن شبکه داده شود. در این مثال، اولین زیرشبکه از آدرسهایی استفاده می‌کند که مثلاً از 130.50.4.1 شروع می‌شوند. آدرسهای دومین زیرشبکه از 130.50.8.1 و آدرس سومین زیرشبکه از 130.50.12.1 شروع می‌شوند. برای آنکه ببینیم چرا شمارهٔ زیرشبکه‌ها چهار تا چهار تا اضافه می‌شود، شکل دودویی این آدرسها را در نظر می‌گیریم:

زیرشبکهٔ ۱	10000010	00110010	000001|00	00000001
زیرشبکهٔ ۲	10000010	00110010	000010|00	00000001
زیرشبکهٔ ۳	10000010	00110010	000011|00	00000001

در الگوی بالا خط عمودی که به صورت | نشان داده شده مرز بین شمارهٔ زیرشبکه و شمارهٔ ماشین را مشخص می‌نماید. در سمت راست، ۶ بیت برای شمارهٔ زیرشبکه و درست چپ ۱۰ بیت برای شمارهٔ ماشین در نظر گرفته شده است.

بررسی عملکرد زیرشبکه‌ها مستلزم آنست که تشریح کنیم بسته‌های IP در مسیریابها چگونه پردازش می‌شوند. هر مسیریاب جدولی را در اختیار دارد که در آن مجموعه‌ای از آدرسهای IP به صورت زوجهای (۰ , **شمارهٔ شبکه**) و (**شمارهٔ ماشین , همین شبکه**)[1] درج شده است. نوع اول مشخص می‌کند که چگونه می‌توان به یک شبکهٔ راه دور رسید. نوع دوم، آدرس ماشینهای محلی را [که مستقیماً به آن مسیریاب متصلند] مشخص می‌کند. همچنین در این جدول، آدرس کارت واسط شبکه برای رسیدن به هر یک از این زیرشبکه‌ها (و پاره‌ای اطلاعات اضافی) درج شده است.

وقتی یک بستهٔ IP توسط یک مسیریاب دریافت می‌شود، آدرس مقصد آن در جدول مسیریابی جستجو می‌شود. اگر مقصد بسته یک شبکهٔ راه دور باشد از طریق یکی از کارتهای واسط به سوی مسیریاب بعدی هدایت می‌شود. اگر مقصد بسته یکی از ماشینهای محلی باشد (مثلاً یکی از ماشینهای LAN متصل به مسیریاب)، مستقیماً برای آن ماشین ارسال می‌شود. اگر مقصد بسته وجود نداشته یا شناسایی نشود، بسته به سوی یک مسیریاب پیش فرض (Default Router) هدایت می‌شود. این مسیریاب دارای جدول مسیریابی کاملتر بوده و احتمالاً یک «مسیریاب مرزی» است. پس طبق این الگوریتم، هر مسیریاب فقط باید مشخصات دیگر شبکه‌ها (یا به عبارت بهتر زیرشبکه‌ها) و همچنین ماشینهای محلی خود را بداند و بدین ترتیب اندازهٔ جدول مسیریابی بشدت کاهش می‌یابد.

وقتی زیرشبکهٔ جدیدی تعریف می‌شود، جدول مسیریابی تغییر می‌کند و در آن درایه‌هایی به شکل (0 , **شمارهٔ زیرشبکه , شمارهٔ همین شبکه**) و (0 , **شمارهٔ همین زیرشبکه , شمارهٔ همین شبکه**)[2] اضافه می‌شود. بدین ترتیب یک مسیریاب که به زیرشبکه k متصل است فقط می‌داند که چگونه باید به زیرشبکه‌های دیگر برسد و همچنین آدرس تمام ماشینهای متصل به زیرشبکه خودش (یعنی زیرشبکه k) را می‌داند. بنابراین مسیریابها مجبور نیستند جزئیات آدرس ماشینهای واقع در زیرشبکه‌های دیگر را بدانند. در حقیقت تنها کاری که مسیریاب در برخورد با آدرسهای IP متعلق به ماشینهای خارج از زیرشبکه خود انجام می‌دهد آنست که: آدرسهای IP را با «**الگوی زیرشبکه**» (Subnet Mask) به صورت بولیَن AND می‌کند تا بخش شمارهٔ ماشین آن حذف شود. سپس آدرس حاصل را در جدول خود جستجو می‌کند. به عنوان مثال فرض کنید بسته‌ای به آدرس 130.50.15.6 ارسال و توسط مسیریاب مرکزی (شکل ۵.۵۷) دریافت شده باشد. مسیریاب مرکزی این آدرس را با الگوی زیرشبکه 255.255.252.0/22 به صورت بولیَن AND می‌کند و نتیجهٔ 130.50.12.0 بدست می‌آید. این آدرس در جدول مسیریابی جستجو می‌شود تا خط خروجی مناسب برای رسیدن به زیرشبکه ۳ مشخص گردد.

تعریف زیرشبکه در حقیقت با ایجاد یک سلسله مراتب سه‌سطحی حجم جدول مسیریابی را کاهش می‌دهد. این سلسله مراتب سه‌سطحی تشکیل شده از: **شمارهٔ شبکه + شمارهٔ زیرشبکه + شمارهٔ ماشین میزبان.**[3]

---

۱. (Network, 0)　　(this-network, host)

۲. (this-network, subnet, 0)　　(this-network,this subnet, host)

۳. مسیریابی سلسله‌مراتبی را در بخش ۵-۲-۶ از همین فصل مطالعه کنید.

CIDR[1]: مسیریابی براساس آدرسهای بدون کلاس

چندین دهه است که از پروتکل IP استفادهٔ گسترده‌ای می‌شود. عملکرد این پروتکل بسیار عالی بوده و همانگونه که اشاره کردیم به رشد نمایی شبکهٔ اینترنت کمک کرده است. متأسفانه IP در حال قربانی شدن در پای محبوبیت خود است چراکه با کمبود فضای آدرس مواجه شده است. از زمانی که افق این مشکل پدیدار شد، بحثها و مناقشات بسیار گسترده‌ای را در جامعهٔ اینترنت دامن زد که چه کاری می‌توان برای آن انجام داد. در بخش جاری، این مشکل و راه حلهای پیشنهادی را تشریح می‌نماییم.

تا سال ۱۹۸۷ افراد دوراندیشی که حدس می‌زدند در روزگاری تعداد شبکه‌های متصل به اینترنت به مرز ۱۰۰,۰۰۰ برسد، انگشت‌شمار بودند و حتی اغلب متخصصین این فنّ نیز چنین ایده‌ای را به مسخره می‌گرفتند ولی صدهزارمین شبکهٔ دنیا در سال ۱۹۹۶ به اینترنت پیوست و به نحوی که در بالا اشاره شد اینترنت در حال مواجه شدن با کمبود آدرسهای IP است. در اصل بیش از دو میلیارد آدرس IP وجود دارد ولی در عمل به دلیل تقسیم‌بندی نامناسب فضای آدرس در قالب چند کلاس، میلیونها آدرس IP به هدر رفته است. (شکل ۵-۵۵ را ببینید.) بالاخص، بیشترین محدودیت در کلاس B است. برای بسیاری از مؤسسات یک شبکهٔ کلاس A با شانزده میلیون آدرس، بسیار بزرگ و بی‌مصرف و کلاس C با ۲۵۶ آدرس بسیار کوچک و ناکافی است؛ فقط شبکهٔ کلاس B با ۶۵۵۳۶ آدرس مناسب است. این مشکل در فرهنگ رایج اینترنت به نام مشکل «سه خرس» مشهور شده است. (برگرفته از داستان موطلایی و سه خرس)

حقیقت آنست که حتی کلاس B هم برای بسیاری از سازمانها و مؤسسات، بسیار بزرگ است. بررسیها نشان داده که بیش از نیمی از شبکه‌های کلاس B، کمتر از ۵۰ ماشین دارند! برای چنین شبکه‌هایی استفاده از کلاس C کفایت می‌کند ولی بی‌تردید تمام مؤسساتی که متقاضی کلاس B بوده‌اند بدان می‌اندیشیده‌اند که روزی فیلد هشت بیتی کلاس C برای شبکهٔ آنها کافی نخواهد بود. نگاهی به گذشته نشان می‌دهد که بهتر آن بوده در کلاس C فیلد شمارهٔ شبکه به جای هشت بیت، ده بیت می‌بود تا در هر شبکه ۱۰۲۲ ماشین میزبان قابل آدرس‌دهی باشد. اگر چنین حالتی را می‌داشتیم شاید بسیاری از مؤسسات و سازمانها به کلاس C اکتفا می‌کردند و در عین حال نیم میلیون از چنین شبکه‌هایی قابل تعریف بود. (برخلاف کلاس B که فقط ۱۶۳۸۴ شبکه قابل تعریف است).

نمی‌توان تقصیر را به گردن طراحان اینترنت انداخت که چرا تعداد کلاسهای B را بیشتر (و با فضای آدرس کوچکتر) در نظر نگرفتند. در روزگاری که تصمیم گرفته شد فقط سه کلاس وجود داشته باشد، اینترنت شبکه‌ای تحقیقاتی بود که فقط مراکز پژوهشی و دانشگاهی مهم ایالات متحده را بهم متصل می‌کرد. (البته به اضافهٔ معدود شرکتها و سایتهای نظامی که آنها نیز در کار پژوهش بودند.) هیچکس نمی‌توانست پیش‌بینی کند که اینترنت روزی بتواند در حد وسیع و به عنوان سیستمی ارتباطی حتی با شبکه‌های تلفن رقابت نماید! در آن زمان هیچکس در این ادعا شکی نداشت که اگر تمام ۲۰۰۰ کالج و دانشگاه ایالات متحده و حتی اکثر دانشگاههای جهان به اینترنت بپیوندند باز هم ۱۶۰۰۰ تا نمی‌شود چراکه این تعداد دانشگاه در کل دنیا وجود نداشت. به علاوه در آن زمان برای آنکه پردازش بسته‌ها سریعتر انجام بشود بهتر بود فیلد شمارهٔ ماشین در آدرس IP، تعداد صحیحی از بایت باشد. [به دلیل سرعت پایین پردازنده‌ها در آن زمان، پردازش بیتی فیلدهای آدرس، از سرعت مسیریابی می‌کاست.]

ولیکن در طرف مقابل اگر مثلاً برای فیلد شمارهٔ شبکه در کلاس B، بیست بیت کنار گذاشته می‌شد مشکل دیگری بروز می‌کرد: «مشکل رشد انفجاری جداول مسیریابی». از دید مسیریابها فضای آدرسهای IP، سلسله مراتب دوسطحی شامل شمارهٔ شبکه و شمارهٔ ماشینها است. مسیریابها مجبور به دانستن شمارهٔ ماشینهای میزبان

1. CIDR : Classless InterDomain Routing

نیستند ولی باید شمارهٔ شبکه‌ها را بدانند. اگر حتی نیمی از شبکه‌های کلاس C بکار گرفته شده باشند هر مسیریاب در کل اینترنت نیاز به جدولی با نیم میلیون درایه (Entry) دارد تا بتواند خط خروجی مناسب برای رسیدن به هر شبکه را مشخص نماید. (گذشته از اطلاعات دیگری که به ازای هر شبکه باید در جدول مسیریابی درج شود.)

شاید تهیهٔ فضای فیزیکی لازم برای ذخیرهٔ نیم میلیون درایه (Entry) در جدول مسیریابی، امکان‌پذیر باشد هر چند برای مسیریابهایی که جداول مسیریابی را در حافظهٔ نوع ایستا (Static RAM) ذخیره می‌کنند، چنین فضایی بسیار گران تمام می‌شود. مشکل اساسی در پیچیدگی الگوریتمهای مدیریت و پردازش چنین جدولی است. پیچیدگی زمانی این الگوریتمها غیرخطی است.[۱] از این بدتر آنکه نرم‌افزار یا سخت‌افزار طراحی شده برای مسیریابهای موجود، زمانی طراحی شده که به اینترنت بیش از هزار شبکه متصل نبود و به نظر می‌رسید که یک دهه طول می‌کشد تا این تعداد به ۱۰۰۰۰ برسد. امروزه اینگونه طراحیها بهینه نیستند.

مضاف بر این، در الگوریتمهای مختلف مسیریابی نیاز است که جداول مسیریابی بطور متناوب ارسال و مبادله شود. (مثل الگوریتم بردار فاصله) هر چه جداول مسیریابی بزرگتر باشد احتمال آنکه بخشی از آن در حین مبادله از دست برود بیشتر خواهد شد و به نقص داده‌های جدول مسیریابی و احتمالاً ناپایداری فرآیند هدایت بسته‌ها خواهد انجامید.

مشکل جداول مسیریابی را می‌توان با افزایش «سطوح سلسله‌مراتب» حلّ کرد. مثلاً می‌توان آدرسهای IP را بدین نحو تعریف کرد که شامل فیلدهای کشور، ایالت / استان، شهر، شبکه و شمارهٔ ماشین میزبان باشد. بدین ترتیب مسیریابهای ستون فقرات در جهان فقط باید در خصوص مسیرهای رسیدن به هر کشور آگاهی داشته باشند، مسیریابهای درون کشور در خصوص مسیرهای رسیدن به هر ایالت یا استان، مسیریابهای ایالت یا استان در مورد مسیرهای رسیدن به هر شهر و مسیریابهای هر شهر فقط باید راه رسیدن به هر شبکه را بدانند. متأسفانه چنین راه حلی نیازمند فضایی بزرگتر از ۳۲ بیت برای آدرس IP است و طبعاً از فضای آدرس استفادهٔ بهینه نخواهد شد. (چرا که مثلاً در این الگو کشور لیختن اشتاین به همان تعداد بیت در آدرس IP دارد که کشور ایالات متحده!)

کوتاه سخن آنکه هر یک از راه حلهای ارائه شده مشکلی را حل و مشکل جدیدی را ایجاد می‌کردند. راه حل نهایی و پیاده شده در اینترنت که توانست به اینترنت اجازهٔ نفس کشیدن بدهد، روش CIDR (مسیریابی براساس آدرسهای بدون کلاس) بود. ایدهٔ اصلی در CIDR که در سند RFC 1519 تشریح شده آنست که آدرسهای IP بدون در نظر گرفتن کلاس و به صورت بلوکهایی با طول متغیر تخصیص یابد. مثلاً اگر یک سایت نیاز به ۲۰۰۰ آدرس داشته باشد یک بلوک آدرس ۲۰۴۸ تایی به او داده می‌شود.

حذف کلاسهای آدرس، فرآیند هدایت بسته‌ها را پیچیده‌تر می‌کند. در سیستم مبتنی بر کلاس، فرآیند هدایت بدین نحو بود که وقتی بسته‌ای به یک مسیریاب می‌رسید، یک کپی از آدرس IP به اندازهٔ ۲۸ بیت به راست شیفت داده می‌شد تا فقط چهار بیت سمت چپ آدرس (که کلاس آدرس را مشخص می‌کند) باقی بماند. براساس این چهار بیت (۱۶ حالت مختلف) بسته‌ها در یکی از کلاسهای A و B و C (و D در صورت پشتیبانی از آن) مرتب می‌شدند. (از این ۱۶ حالت مختلف، هشت حالت برای کلاس A است ـ 0xxx ـ چهار حالت برای کلاس - 10xx - B ، دو حالت برای کلاس C - 110x - و دو حالت برای کلاسهای D و E است.) پس از تشخیص کلاس آدرس، برای بدست آوردن شمارهٔ شبکه، آدرس IP با یکی از الگوهای 8- ، 16- ، 24- به صورت بولی AND و بخش شمارهٔ ماشین حذف می‌شد. سپس شمارهٔ شبکه در هر یک از جداول مربوط به آدرسهای کلاس A ، B و C

---

۱. پیچیدگی غیرخطی این الگوریتمها عموماً $O(n^2)$ و $O(n.logn)$ است. ـ م

جستجو می‌شد. جداول مسیریابی برای کلاسهای A و B برحسب شمارهٔ شبکه ایندکس شده بودند.[۱] در عوض جدول مسیریابی برای کلاس C مبتنی بر روش جداول Hash (Hash Table) پیاده شده بود. پس از آنکه درایهٔ متناظر با آدرس شبکه در یکی از این جداول پیدا می‌شد خط خروجی متناسب با آن شبکه مشخص شده و بسته بر روی آن خط هدایت می‌گردید.

در CIDR این الگوریتم ساده، کار نخواهد کرد. در عوض به هر یک از درایه‌های جدول مسیریابی یک فیلد ۳۲بیتی جدید افزوده شده که الگوی آدرس را [از طریق یک MASK سی و دوبیتی] مشخص می‌کند. بدین ترتیب برای تمام شبکه‌ها فقط یک جدول مسیریابی یکتا وجود دارد که در حقیقت یک آرایهٔ سه ستونی متشکل از آدرس IP، الگوی زیرشبکه (Subnet Mask) و خط خروجی است. وقتی بسته‌ای وارد می‌شود ابتدا آدرس IP آن استخراج می‌شود. سپس جدول مسیریابی درایه به درایه (Entry by Entry) جستجو و آدرس مقصد بسته پس از AND شدن با الگوی زیرشبکه از هر درایه با آدرس IP از آن درایه مقایسه می‌شود. این فرآیند آنقدر تکرار می‌گردد تا به موارد مطابقت برسد. این امکان وجود دارد که چندین درایه با یک آدرس IP مطابقت داشته باشد (به دلیل طول متفاوت الگوهای زیرشبکه). در این حالت درایه‌ای که طول الگوی زیرشبکه آن از همه بزرگتر است از بین آنها انتخاب می‌شود. به عبارتی اگر دو مورد تطابق با طول الگوی 20/ (255.255.240.0) و الگوی 24/ (255.255.255.0) پیدا شود، درایهٔ دوم انتخاب می‌شود.

برای سرعت بخشیدن به فرآیند جستجو و مطابقت، الگوریتمهای پیچیده‌ای ابداع شده است. (Ruiz-Sanches et al. 2001) مسیریابهای تجاری در بازار امروز از تراشه‌های VLSI خاصی بهره گرفته‌اند که الگوریتم مذکور را به صورت یک «سخت‌افزار درون‌کار» (Embedded Hardware) پیاده‌سازی کرده‌اند.

برای آنکه فهم فرآیند هدایت بسته‌ها در CIDR را ساده‌تر کنیم مثالی را مدنظر قرار بدهید که در آن میلیونها آدرس تعریف شده است و آدرس شروع 194.24.0.0 است. فرض کنید که دانشگاه کمبریج به ۲۰۴۸ آدرس نیاز دارد و آدرسهای 194.24.0.0 تا 194.24.7.255 به آن اختصاص داده شده است. (الگوی زیرشبکه نیز 255.255.248.0 است.) بعداً دانشگاه آکسفورد تقاضای ۴۰۹۶ آدرس IP می‌دهد. از آنجایی که بلوکهای آدرس ۴۰۹۶ تایی باید در مرز ۴۰۹۶ بایتی قرار بگیرد نمی‌توان آدرسهایی که از 194.24.8.0 شروع می‌شود را به آن اختصاص داد. در عوض آدرس اختصاص داده شده به او در محدودهٔ 194.24.16.0 تا 194.24.31.255 و با الگوی 255.255.240.0 خواهد بود. در اینجا دانشگاه ادینبورو تقاضای ۱۰۲۴ آدرس داده و فضای 194.24.8.0 تا 194.24.11.255 با الگوی 255.255.252.0 به او تعلق می‌گیرد. این انتسابها در جدول ۵-۵۹ خلاصه شده‌اند.

دانشگاه	اولین آدرس	آخرین آدرس	تعداد آدرس	الگوی نمایش
Cambridge	194.24.0.0	194.24.7.	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
در دسترس و آزاد	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

شکل ۵-۵۹. انتساب آدرسهای IP.

۱. به عبارت ساده به دلیل کم بودن تعداد شبکه‌ها جداول مسیریابی برای کلاسهای A و B در ساختمان داده‌ای شبیه به‌آرایه ذخیره می‌شد. ـم

حال جداول مسیریابی در تمام مسیریابهای واقع بر ستون فقرات اینترنت در جهان باید با این سه درایهٔ جدید به‌هنگام شود. هر درایه دارای یک آدرس مبنا و یک الگوی زیرشبکه است. این درایه‌ها در مبنای دو عبارتند از:

	آدرس	الگوی زیرشبکه (Subnet Mask)
C:	11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E:	11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
O:	11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

حال ببینیم وقتی که بسته‌ای با آدرس 194.24.17.4 وارد یک مسیریاب می‌شود چه اتفاقی می‌افتد. این آدرس به صورت دودویی عبارت است از:

11000010 00011000 00010001 00000100

ابتدا این آدرس با الگوی زیرشبکه کمبریج، AND می‌شود و نتیجهٔ زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار با آدرس مبنای دانشگاه کمبریج مطابقت ندارد. حال مجدداً آدرس اصلی با الگوی زیرشبکه دانشگاه ادینبورو AND شده و نتیجهٔ زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار نیز با آدرس مبنای دانشگاه ادینبورو تطابق ندارد و همین کار برای دانشگاه آکسفورد تکرار شده مقدار زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار با آدرس مبنای دانشگاه آکسفورد مطابقت دارد. اگر هیچ مورد تطبیق دیگری در جدول یافت نشد بسته بر روی خطی ارسال می‌شود که در درایهٔ متناظر با شبکهٔ دانشگاه آکسفورد درج شده است.

حال اجازه بدهید، آدرس این سه دانشگاه را از دید یک مسیریاب در نبراسکای اوهاما بررسی کنیم. این مسیریاب چهار خط به میناپولیس، نیویورک، دالاس و دِنور دارد. وقتی نرم‌افزار مسیریاب اوهاما، این سه درایهٔ جدید را جهت درج در جدول مسیریابی خود دریافت می‌دارد، متوجه می‌شود که قادر است هر سه تای آنها را در یک «درایهٔ واحد و تجمیع شده» (Aggregate Entry) به صورت 194.24.0.0/19 ادغام نماید.[1] آدرس و الگوی زیرشبکه در مبنای دو به صورت زیر است:

11000010 00000000 00000000 00000000 11111111 11111111 11100000 00000000

طبق این درایه تمام بسته‌هایی که به مقصد یکی از این سه دانشگاه روانه شده‌اند به سوی نیویورک هدایت می‌شوند. با تجمیع این سه درایه، مسیریاب اوهاما توانسته به میزان دو درایه حجم جدول خود را کاهش بدهد.

به همین ترتیب اگر مسیریاب نیویورک برای تمام ترافیک منتهی به انگلستان فقط یک خط به لندن داشته باشد او نیز سه درایهٔ فوق را در یک درایه ادغام می‌کند ولیکن اگر برای لندن و ادینبورو دو خط مجزا داشته باشد باید هر سه‌تای آنها را بطور مجزا در جدول خود ذخیره کند. عمل تجمیع (Aggregation) در اینترنت به طور گسترده‌ای مورد استفاده قرار گرفته تا حجم جداول مسیریابی کاهش یابد.

آخرین نکته در این مثال آن است که بر طبق درایهٔ ادغام شده در جدول مسیریابی مسیریاب واقع در اوهاما

۱. از آن جهت امکان تجمیع این سه آدرس وجود داشته که بسته‌هایی که مقصدشان هریک از این سه دانشگاه است باید بر روی خط خروجی یکسانی بروند. -م

حتی بسته‌هایی که به آدرس اختصاص داده نشده روانه هستند [یعنی آدرسهای بین 194.24.12.0 تا 194.24.15.255] نیز به سوی نیویورک هدایت می‌شوند. مادامی که این آدرسها به کسی اختصاص داده نشده هیچ مشکلی به وجود نمی‌آید چرا که بنا نیست بسته‌هایی با این آدرسها تولید شوند. ولی اگر این بلوک آدرس ، به شرکتی در کالیفرنیا داده شود باید درایه‌ای جدید به شکل 194.24.12.0/22 در جداول مسیریابی تمام مسیریابها درج شود تا بسته‌هایی به مقصد این شبکه نیز بدرستی مسیریابی شوند.

NAT[1] : ترجمهٔ آدرسهای شبکه

آدرسهای IP کمیاب و ارزشمند هستند: یک ISP ممکن است یک بلوک آدرس با الگوی 16/ (همان کلاس B سابق) و توانایی آدرس‌دهی ۶۵۵۳۴ ماشین میزبان، داشته باشد. اگر تعداد مشتریان این ISP از این تعداد بیشتر شود مشکل بهم می‌زند. برای مشتریان خانگی که از طریق خطوط تلفن متصل می‌شوند، راه حل این مشکل آن است که وقتی مشتری شماره‌گیری کرد و وارد شد به او موقتاً یک آدرس IP پویا اختصاص داده شود و پس از پایان نشست و قطع ارتباط، این آدرس پس گرفته شود. در این روش شبکه‌ای با آدرس 16/ (کلاس B) می‌تواند حداکثر ۶۵۵۳۴ کاربر فعّال داشته باشد که این تعداد حتی برای یک ISP با چند صدهزار مشتری نیز کفایت می‌کند. به محض آنکه یک نشست خاتمه یافت آدرس IP منتسب شدهٔ قبلی، به تماس گیرندهٔ بعدی داده می‌شود. این استراتژی اگرچه برای یک ISP با تعداد متوسطی از کاربران خانگی به خوبی کار می‌کند ولی برای ISPهایی که به مشتریان اداری خدمات می‌دهند مفید نیست.

مشکل از اینجا ناشی می‌شود که مشتریان اداری انتظار دارند که حداقل در ساعات کاری روز خطی دائم و فعال (On-Line) داشته باشند. امروزه، چه دفاتر کوچک اداری مثل یک آژانس مسافرتی با سه کارمند و چه شرکتهای بزرگ که دارای تعداد زیادی کامپیوتر و شبکهٔ محلی هستند، نیاز به خط دائم و فعال دارند. برخی از این کامپیوترها، PC کارمندان و برخی دیگر مثلاً سرویس دهنده‌های وب هستند. عموماً در هر LAN یک مسیریاب وجود دارد که از طریق یک خط اجاره‌ای (Leased) به ISP متصل شده است. چنین ساختاری متضمن آن است هر کامپیوتر آدرس IP خود را داشته باشد و به طور روزانه تغییر نکند. در نتیجه، تعداد کل کامپیوترهایی که در اختیار مشتریان اداری است نباید از تعداد آدرسهای IP متعلق به ISP بیشتر شود. برای آدرس 16/، حداکثر تعداد کامپیوترها ۶۵۵۳۴ است. برای یک ISP با دهها هزار مشتری اداری، این فضا سریعاً اشباع می‌شود.

آنچه که مشکل را حادّتر می‌کند آن است که روزبه‌روز بر تعداد مشترکین اینترنت از طریق مودمهای کابلی ADSL افزوده می‌شود. ویژگی چنین سرویسی عبارت است از: (۱) کاربر یک آدرس IP دائم و ثابت می‌گیرد. (۲) هزینهٔ شماره‌گیری و اتصال ندارد (مگر یک هزینهٔ ثابت ماهانه) بدین ترتیب اینگونه کاربران همیشه در شبکه حضور دارند. این موضوع، مشکل کمبود آدرسهای IP را افزایش می‌دهد. در اینجا تخصیص موقت آدرسهای IP (شبیه به مکانیزمی که برای کاربران تلفنی داشتیم) عملی نیست.

حتی از این هم پیچیده‌تر آنکه ممکن است کاربران ADSL و اینترنت کابلی دارای دو یا چند کامپیوتر در خانه باشند و تمام اعضای خانواده از طریق همین خط فعال و مشترک به ISP متصل شوند. یک راه حل آن است که تمام PCها از طریق یک LAN به هم متصل شده و با یک مسیریاب به ISP وصل شوند. از دیدگاه ISP شبکهٔ این خانواده فرقی با یک دفتر اداری کوچک ندارد.

مشکل کمبود آدرسهای IP یک مسئله تئوریک نیست که در آینده‌ای دور رخ بدهد. همین الان با این مشکل مواجه هستیم. راه حل طولانی مدت و همیشگی این مشکل آنست که به سوی IPv6 (که آدرسهای آن ۱۲۸ بیتی

---
۱. Network Address Translation

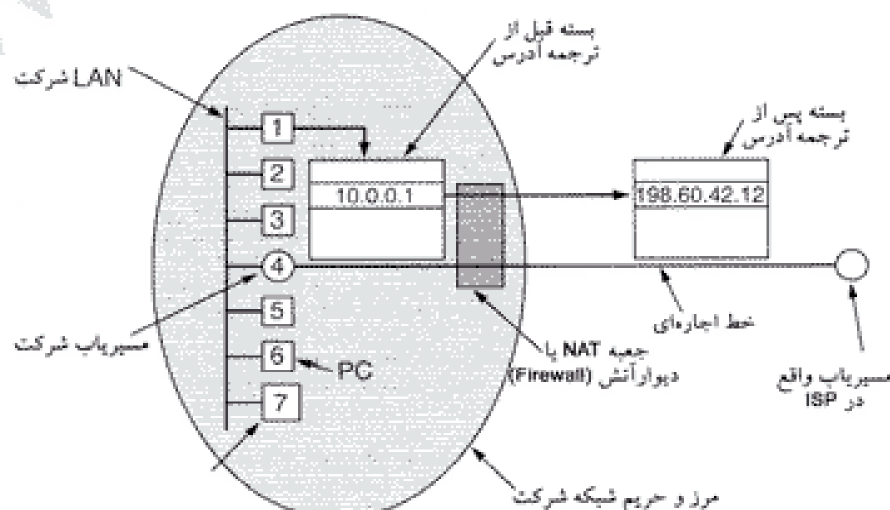
است) حرکت نماییم ولیکن گذار از نسخهٔ ۴ به نسخهٔ ۶ به آهستگی صورت می‌گیرد و سالها طول می‌کشد تا این تغییر به طور کامل انجام شود. در نتیجه بسیاری افراد احساس کردند که به یک راه حل سریع و کوتاه مدت نیاز است. این راه حل سریع، NAT (ترجمهٔ آدرس شبکه) است که در RFC 3022 تشریح شده و در ادامه آنرا مختصراً بررسی می‌نماییم. برای کسب آگاهی بیشتر به مرجع (Dutcher, 2001) مراجعه نمایید.

ایدهٔ اصلی در NAT آن است که به هر شرکت یک یا تعداد کمی آدرس IP معتبر و جهانی اختصاص بدهیم. درون این شرکت، هر کامپیوتر دارای یک آدرس IP یکتا است که برای مسیریابی ترافیک داخلی بکار می‌آید. با این حال وقتی بسته‌ای بخواهد شرکت را ترک کرده و به ISP برود باید قبل از خروج ترجمهٔ آدرس صورت بگیرد. برای آنکه این روش ممکن باشد سه محدوده از فضای آدرس IP جهت بکارگیری در شبکه‌های داخلی، به صورت «خصوصی» (Private) تعریف شده است و شرکتها می‌توانند به صورت دلخواه از آنها استفاده کنند. [نیازی به ثبت جهانی آنها نیست.] تنها قانون آن است که هیچ بسته‌ای نباید با چنین آدرسی بر روی اینترنت ظاهر شود. این سه محدودهٔ رزرو شده عبارتند از:

10.0.0.0	- 10.255.255.255/8	(16,777,216 Hosts)
172.16.0.0	- 172.31.255.255/12	(1,048,576 Hosts)
192.168.0.0	- 192.168.255.255/16	(65535 Hosts)

در محدودهٔ اول ۱۶۷۷۷۲۱۶ آدرس (به استثنای ۰ و ۱-) در دسترس است و عموماً اکثر شرکتها از آن استفاده می‌کنند هر چند نیازی به چنین تعداد آدرسی نداشته باشند.

عملکرد NAT در شکل ۵-۶۰ نشان داده شده است. ماشینها در درون شبکه دارای یک آدرس یکتا به فرم 10.x.y.z هستند. ولیکن وقتی بسته‌ای بخواهد مرز شرکت را ترک کند ابتدا باید از درون یک «جعبهٔ NAT» (NAT BOX) عبور کرده و آدرس مبداء آن با آدرس IP حقیقی شرکت جانشین شود. مثلاً در شکل ۵-۶۰ آدرس 10.0.0.1 با آدرس 198.60.42.1 عوض شده است. اغلب «جعبهٔ NAT» در یک «دیوار آتش» (Firewall) ادغام می‌شود تا این ابزار ضمن ترجمهٔ آدرس، امنیت شبکه را نیز با نظارت دقیق بر ورود و خروج اطلاعات تضمین نماید. در فصل ۸ مفهوم «دیوار آتش» را بررسی خواهیم کرد. همچنین می‌توان «جعبهٔ NAT» را در مسیریاب شرکت قرار داد. اکثر مسیریابهای امروزی از فرآیند NAT پشتیبانی می‌کنند.



شکل ۵-۶۰. مکان و عملکرد «جعبهٔ NAT»

تا اینجا جزئیات کمی از فرآیند NAT مطرح کرده‌ایم: وقتی پاسخ یک بسته بر می‌گردد (مثلاً از سرویس‌دهندهٔ وب) طبعاً آدرس ماشین گیرندهٔ پاسخ، 192.60.42.1 است. سؤال این است که جعبهٔ NAT از کجا بداند که آدرس کدام ماشین داخلی را به جای آن قرار بدهد؟ مسئلهٔ اصلی در NAT همین نکته است. اگر فیلدی اضافی در سرآیند بستهٔ IP وجود داشت می‌شد از آن برای درج آدرس واقعی گیرندهٔ بسته بهره گرفت ولیکن در سرآیند بسته تنها یک بیت بلااستفاده مانده است. همچنین می‌توان یک گزینهٔ جدید [در فضای فیلد اختیاری Option] تعریف کرد تا آدرس حقیقی ماشین مبداء بسته را نگاه دارد ولی انجام این کار مستلزم آن است که کُد نرم‌افزار IP در تمام ماشینها و در کل اینترنت تغییر کند تا گزینهٔ جدید به رسمیت شناخته شده و به درستی تعبیر شود. این راه حل نیز فرآیند زمان‌بری است و مشکل را در کوتاه مدت حل نخواهد کرد.

آنچه که بطور واقعی اتفاق می‌افتد به نحو ذیل است: طراحان NAT بدین نتیجه رسیده بودند که اغلب بسته‌های IP در درون فیلد دادهٔ خود یک بستهٔ TCP یا UDP حمل می‌کنند. هرگاه در فصل ششم TCP و UDP را بررسی کردیم، خواهید دید که هر دوی این پروتکلها دارای سرآیندی برای بسته‌های خود هستند که دو فیلد **«شمارهٔ پورت مبداء»** و **«شمارهٔ پورت مقصد»** جزو آنهاست. در زیر اگرچه تمرکز ما بر پورتهای TCP است ولی همین قضیه برای پورتهای UDP نیز صادق است. این پورتها که اعداد صحیح ۱۶ بیتی هستند، مشخص می‌کنند که اتصال TCP از چه پروسه‌ای شروع و به چه پروسه‌ای ختم می‌شود. این شمارهٔ پورتها فیلدهای لازم برای عملکرد NAT را فراهم آورده‌اند.

هرگاه یک پروسه بخواهد یک اتصال TCP با یک پروسهٔ راه دور برقرار کند یک شمارهٔ پورت بلااستفاده برای خود بر می‌گزیند. این پورت، اصطلاحاً «پورت مبداء» نام دارد و به کد برنامه TCP تفهیم می‌کند که باید بسته‌های ورودی با این شمارهٔ پورت را برای او بفرستد. همچنین هر پروسه یک شمارهٔ پورت مقصد تعیین می‌کند تا مشخص شود که بسته‌ها باید به کدام پروسه در ماشین مقصد تحویل شود. شمارهٔ پورتهای صفر تا ۱۰۲۳ برای سرویس دهنده‌های مشهور رزرو شده است. به عنوان مثال پورت شمارهٔ ۸۰ توسط سرویس دهنده‌های وب بکار گرفته شده است، لذا برنامه‌های مشتری (Client) براحتی با آنها ایجاد ارتباط می‌کنند. کوتاه سخن آنکه، هر پیام خروجی از TCP دارای شمارهٔ پورت مبداء و شماره پورت مقصد است و این دو شمارهٔ پورت هویت پروسه‌های طرفین ارتباط را مشخص می‌نمایند.

تمثیلی از یک نمونه می‌تواند به فهم شماره‌های پورت کمک کند: یک شرکت را در نظر بگیرید که دارای یک شماره تلفن اصلی و واحد است. وقتی افراد با این شماره تماس می‌گیرند بلافاصله اپراتور مربوطه از آنها سؤال می‌کند که کدام شمارهٔ داخلی مدّنظر آنهاست؛ سپس خط داخلی را وصل می‌کند. شمارهٔ اصلی به مثابهٔ آدرس IP است و شماره‌های داخلی، مشابه با شماره پورت هستند. پورتها، ۱۶ بیت آدرس اضافی دیگر هستند که هویت پروسهٔ گیرندهٔ بسته‌ها را مشخص می‌نمایند.

با استفاده از فیلد «شمارهٔ پورت مبداء» می‌توان مشکل نگاشت آدرسها در NAT را حل کرد. هر گاه بسته‌ای برای خروج از شبکه به NAT وارد شود، آدرس مبداء آن که به شکل 10.x.y.z است با آدرس IP حقیقی و معتبر شرکت عوض می‌شود. مضاف بر این فیلد شمارهٔ پورت مبداء (TCP Source Port) با عددی عوض می‌شود که در حقیقت این عدد اندیس جدول ترجمهٔ آدرس در جعبهٔ NAT است. هر یک از درایه‌های این جدول، آدرس IP اصلی و همچنین شمارهٔ پورت واقعی آن بسته را نگه می‌دارند. در آخر کد کشف خطای بستهٔ TCP و بستهٔ IP از نو محاسبه و در بسته قرار داده می‌شود. [چرا که هم فیلد شمارهٔ پورت و هم فیلد آدرس IP مبداء در جعبهٔ NAT عوض می‌شود. -م] عوض کردن مقدار فیلد پورت مبداء (Source Port) الزامی است چراکه ممکن است بطور همزمان از دو ماشین به آدرسهای 10.0.0.1 و 10.0.0.2 یک اتصال TCP اتفاقاً با شمارهٔ پورت مبداء یکسان (مثلاً

۵۰۰۰) ایجاد شود، لذا شمارهٔ پورت مبداء نمی‌تواند هویت واقعی پروسهٔ ارسال کنندهٔ بسته‌ها را مشخص کند.[۱]

وقتی بسته‌ای از طریق ISP به جعبهٔ NAT وارد می‌شود ابتدا مقدار فیلد پورت مبداء استخراج شده و از آن به عنوان اندیس جدول نگاشت در جعبهٔ NAT استفاده می‌شود. پس از پیدا شدن درایهٔ متناظر، آدرس IP داخلی و شمارهٔ اصلی پورت مبداء بسته استخراج شده و در درون بسته قرار می‌گیرد. سپس این بسته از طریق مسیریاب داخلی شرکت، برای تحویل به آدرس 10.x.y.z مسیر طبیعی خود را طی می‌کند.

همچنین از NAT می‌توان برای تخفیف مشکل کمبود آدرس IP برای کاربران ADSL و کاربران کابلی بهره گرفت. هر گاه ISP بخواهد به هر یک از این کاربران آدرسی انتساب بدهد، از آدرسی در فضای 10.x.y.z بهره می‌گیرد. قبل از آنکه بسته‌های ماشین کاربران ، ISP را ترک کنند و به اینترنت وارد شوند باید ابتدا وارد جعبهٔ NAT شده و آدرس غیرحقیقی و محلی آنها به آدرس واقعی متعلق به ISP نگاشته شود. در مسیر برگشت، عکس فرآیند نگاشت انجام می‌شود. بدین نحو از دیدگاه اینترنت، این ISP (و کاربران ADSL یا کابلی آن) دقیقاً مثل یک شرکت بزرگ به نظر می‌رسند، هر چند تعداد آدرسهای واقعی و معتبر ISP ناچیز است.

اگرچه این روش مشکل کمبود آدرسهای IP را حل می‌کند ولیکن بسیاری از افراد در جامعه اینترنت از آن به عنوان کاری بی‌ارزش و مردود یاد می‌کنند. برخی از مخالفتهای آنان را به اختصار ارائه می‌نماییم. اول آنکه NAT مدل معماری IP را نقض می‌کند چرا که در این مدل بیان شده که آدرس IP به صورت یکتا ماشینی واحد را در کل جهان مشخص می‌نماید. ساختار تمام نرم‌افزارهای اینترنت با تکیه بر این واقعیت بنیان گذاشته شده است. با NAT ممکن است هزاران ماشین از آدرس 10.0.0.1 استفاده کنند (و می‌کنند).

دوم آنکه NAT اینترنت را از حالت «بدون‌اتصال» به شبکه‌ای «اتصال‌گرا» تبدیل می‌نماید. مسئله اینجاست که جعبهٔ NAT باید اطلاعاتی را در خصوص نگاشت اتصالهایی که از آن می‌گذرند در خود نگاه دارد. نگهداری وضعیت هر اتصال ویژگی شبکه‌های اتصال‌گراست و سنخیتی با شبکه‌های بدون‌اتصال ندارد. اگر جعبهٔ NAT به ناگاه از کار بیفتد و جدول نگاشت آن از دست برود تمام اتصالات TCP برقرار شده از دست می‌رود. بدین ترتیب با وجود NAT ، اینترنت به یک شبکهٔ آسیب‌پذیر مدار مجازی تبدیل می‌شود.

سوم آنکه، NAT اصول بنیانی لایه‌بندی پروتکلها را نقض می‌کند: لایهٔ k نباید هیچ تصوری از آنچه که لایهٔ k+1 در فیلد حمل داده از بستهٔ او قرار می‌دهد، داشته باشد یا در آن دخالتی کند.[۲] اصل اساسی در معماری لایه‌بلایه آنست که تمام لایه‌ها مستقل از دیگری باشند. اگر مثلاً زمانی TCP به نسخهٔ TCP-2 ارتقاء یابد و سرآیند بسته‌ها تغییر کنند (مثلاً شمارهٔ پورتها ۳۲ بیتی شوند)، NAT از کار خواهد افتاد. ایدهٔ اصلی در پروتکلهای لایه‌ای آن بوده که تغییر در یک لایه نیازی به تغییر در لایه‌های دیگر نداشته باشد در حالیکه NAT این عدم وابستگی را از بین می‌برد.

چهارم آنکه پروسه‌های اینترنت مجبور به استفاده از TCP یا UDP نیستند. اگر فرضاً کاربری بر روی ماشین A تصمیم بگیرد برای محاوره و مبادلهٔ داده با کاربری بر روی ماشین B از یک پروتکل جدید در لایهٔ انتقال استفاده کند (مثلاً برای کاربردهای چند رسانه‌ای)، وجود NAT منجر به عدم کارکرد آن برنامهٔ کاربردی خواهد شد چرا که NAT نخواهد توانست فیلد پورت مبداء را به درستی پیدا کرده و از آن استفاده نماید.

پنجم آنکه برخی از برنامه‌های کاربردی آدرس IP ماشین خود را در متن اطلاعات ارسالی قرار می‌دهند. گیرنده نیز این آدرس را استخراج کرده و از آن در جایی استفاده می‌کند. از آنجایی که NAT چیزی در مورد این

---

۱. بعبارت روشنتر چون تمام بسته‌ها در برگشت آدرس IP یکسانی دارند فلذا این آدرس پورت هر بسته است که هویت گیرندهٔ واقعی بسته را مشخص می‌کند و طبعاً NAT باید در هنگام خروج بسته‌ها ضمن عوض کردن شمارهٔ پورت، یکتا بودن آنرا تضمین کند. –م

۲. بعبارتی از دیدگاه لایهٔ k آنچه که از لایهٔ k+1 می‌رسد تعدادی بایت خام است. –م

آدرسهای مخفی نمی‌داند فلذا قادر به تغییر آنها نبوده و هرگونه تلاش برای استفاده از این آدرسهای ناصحیح (در ماشین راه دور) با شکست مواجه می‌شود. FTP ، یعنی استاندارد انتقال فایل در اینترنت به همین ترتیب عمل می‌کند و با وجود NAT از کار می‌افتد مگر آنکه اقدامات احتیاطی خاصی به عمل آید. به دلیل مشابه، پروتکل H.323 که برای تلفن اینترنتی کاربرد دارد (و در فصل هفتم به معرفی آن خواهیم پرداخت) با وجود NAT کار نخواهد کرد. البته می‌توان با تغییرات اصلاحی در NAT آن را بکار گرفت ولی این که با معرفی هر برنامهٔ کاربردی جدید مجبور به اصلاح NAT شویم، اصلاً ایدهٔ جالبی نیست.

ششم آنکه چون فیلد آدرس پورت مبداء، ۱۶ بیتی است، حداکثر ۶۵۵۳۶ ماشین را می‌توان به یک آدرس IP واحد نگاشت. این تعداد حقیقتاً مقدار کمی است (گذشته از آن، ۴۰۹۶ شمارهٔ پورت نیز برای کاربردهای خاص کنار گذاشته شده‌اند)، ولیکن اگر تعداد آدرسهای IP معتبر و جهانی که در اختیار ISP قرار دارد، بیش از یکی باشد به ازای هر یک می‌توان ۶۱۴۴۰ ماشین را با آدرسهای غیرحقیقی مدیریت کرد.

این مشکلات بهمراه مسائل دیگر NAT ، در RFC 2993 تشریح شده است. عموماً مخالفین NAT می‌گویند که با حل نازیبا و موقتی مسئله کمبود آدرسهای IP، اصرار بر روی راه حل واقعی و نهایی که همانا رفتن به طرف IPv6 است، کم می‌شود و تعویق انداختن در این تغییر و تحول اصلاً خوب نیست!

۵-۶-۳ پروتکلهای کنترل اینترنت

اینترنت مضاف بر IP که برای انتقال داده‌ها کاربرد دارد، چندین پروتکل کنترلی دیگر دارد که همگی در لایهٔ شبکه به کار گرفته می‌شوند. این پروتکلها عبارتند از: ICMP ، ARP ، RARP ، BOOTP و DHCP. در این بخش، این پروتکلها را به ترتیب مرور می‌کنیم.

ICMP[1] : پروتکل ارسال پیامهای کنترلی در اینترنت

عملکرد اینترنت توسط مسیریابها به صورت دقیق نظارت و کنترل می‌شود. هر گاه اتفاق غیرمنتظره‌ای بیفتد، رخداد مزبور توسط پروتکل ICMP گزارش می‌شود. این پروتکل همچنین برای آزمایش و رفع عیب شبکه کاربرد دارد. تقریباً یک دوجین از پیامهای ICMP به منظور اطلاع‌رسانی تعریف شده است. مهمترین این پیامها در شکل ۵-۶۱ فهرست شده‌اند. هر پیام ICMP مستقیماً درون یک بستهٔ IP جاسازی و ارسال می‌شود.

نوع پیام	توصیف عملکرد
Destination unreachable	بهر دلیلی بسته را نمی‌توان به مقصد تحویل داد.
Time exceeded	زمان حیات بسته به صفر رسیده است.
Parameter problem	فیلدی از سرآیند بسته مقدار نامعتبر داشته است.
Source quench	بسته دعوت به آرامش
Redirect	حاوی اطلاعاتی در خصوص جغرافیای مسیر و اعلام اشتباه در مسیریابی
Echo	درخواست از یک ماشین تا اگر فعال است پاسخ بدهد.
Echo reply	پاسخ به پیام Echo بمنظور تایید فعالیت
Timestamp request	همانند پیام Echo بهمراه مهر زمان
Timestamp reply	همانند پیام Echo Reply بهمراه مهر زمان

شکل ۵-۶۱. انواع پیامهای اساسی ICMP.

۱. Internet Control Message Protocol

از پیام DESTINATION UNREACHABLE زمانی استفاده می‌شود که زیرشبکه یا مسیریاب نتواند موقعیت مقصد را تشخیص بدهد یا وقتی که بیت DF[1] در بسته‌ای به ۱ تنظیم شده باشد و آن بسته مجبور به عبور از شبکه‌ای باشد که طول بسته‌های آن کوچک است. [طبعاً بسته حذف می‌شود.]

پیام TIME EXCEEDED زمانی ارسال می‌شود که بسته به دلیل صفر شدن شمارندهٔ TTL (شمارندهٔ زمان حیات) حذف گردد. این رخداد نشانهٔ آن است که بسته‌ها در حلقهٔ بی‌نهایت افتاده‌اند یا آنکه ازدحام سنگینی رخ داده یا آنکه مقدار اولیهٔ این شمارنده بسیار کم بوده است.

پیام PARAMETER PROBLEM مشخص‌کنندهٔ آنست که در سرآیند بستهٔ IP مقدار نامعتبر و اشتباهی درج شده است. [لذا مسیریاب آن را حذف کرده و این پیام را ارسال نموده است.] این مشکل از وجود یک اشکال در نرم‌افزار IP ماشین فرستنده یا احتمالاً نرم‌افزار مسیریابهای میانی پرده بر می‌دارد.

پیام SOURCE QUENCH سابقاً برای آن بکار می‌رفته تا به ماشینهایی که بیش از اندازه بسته ارسال می‌کنند اخطار داده شود. هر ماشین که این پیام را دریافت کند باید نرخ ارسال بسته‌های خود را کاهش بدهد. ولیکن از این پیام به ندرت استفاده می‌شود چرا که وقتی ازدحام رخ بدهد، ارسال چنین پیامهایی بر هیزم این آتش می‌افزاید. کنترل ازدحام در اینترنت عمدتاً در لایهٔ انتقال انجام می‌شود. این روش را در فصل ششم خواهیم آموخت.

از پیام REDIRECT زمانی استفاده می‌شود که مسیریاب احساس کند بسته‌ای در مسیر غلط قرار گرفته است. این پیام برای گزارش خطای احتمالی به ماشین مبداء بسته، کاربرد دارد.

پیامهای ECHO و ECHO REPLY بدین منظور کاربرد دارند که ببینیم آیا یک ماشین مقصد در دسترس و فعال است یا خیر. انتظار می‌رود هر ماشینی که پیام ECHO دریافت می‌کند، در پاسخ به آن، پیام ECHO REPLY را برگرداند.

پیامهای TIMESTAMP REQUEST و TIMESTAMP REPLY مشابه با دو پیام قبلی هستند با این تفاوت که در بدنهٔ پاسخ، زمان دریافت پیام و همچنین زمان ارسال پیام درج می‌شود. از این قابلیت می‌توان برای اندازه‌گیری کارآیی شبکه بهره گرفت.

مضاف بر پیامهای فوق‌الذکر، پیامهای دیگری نیز تعریف شده‌اند که می‌توانید فهرست آنها را در آدرس زیر بیابید: www.iana.org/assignments/icmp-parameters

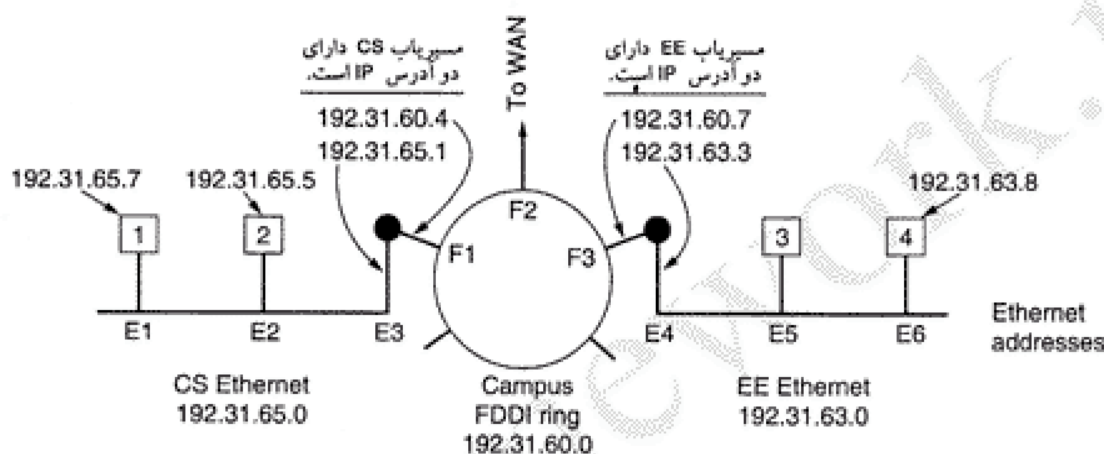
ARP[2] : پروتکل تحلیل آدرس

اگرچه هر ماشین در اینترنت دارای یک (یا چند) آدرس IP است، ولیکن این آدرسها حقیقتاً نمی‌توانند برای ارسال بسته‌ها بکار گرفته شوند چرا که سخت‌افزار لایهٔ پیوند داده‌ها هیچ درکی از آدرسهای IP ندارد. امروزه اغلب ماشینهای میزبان در شرکتها یا دانشگاهها به کمک یک کارت واسط شبکه به LAN متصل شده‌اند و این کارتها فقط آدرسهای MAC (آدرس سخت‌افزاری کارت شبکه) را می‌شناسند. به عنوان مثال هر کارت شبکهٔ اترنت پس از تولید در کارخانه، دارای یک آدرس ۴۸ بیتی یکتا است. سازندگان کارتهای اترنت برای تعبیهٔ آدرس در کارتهای تولیدی خود ابتدا از یک مرکز مجاز تقاضا می‌کنند که یک بلوک آدرس به آنها اختصاص بدهد و بدین ترتیب اطمینان حاصل می‌شود که هیچ دو کارت شبکه در کل جهان دارای آدرس یکسان نیست. (تا از هر گونه برخورد دو کارت شبکه با شماره‌های یکسان بر روی LAN اجتناب شود.) کارتهای شبکه، فریمها را براساس آدرسهای ۴۸ بیتی اترنت، ارسال یا دریافت می‌نمایند و هیچ چیزی در خصوص آدرسهای IP نمی‌دانند.

اکنون این سؤال پیش می‌آید که چگونه آدرسهای IP به آدرسهای لایهٔ پیوند داده (مثل آدرسهای اترنت)

۱. Don't Fragment ۲. Address Resolution Protocol

نگاشته و تبدیل می‌شوند؟ برای تشریح عملیات نگاشت آدرس از مثال شکل ۵-۶۲ استفاده کرده‌ایم. در این شکل یک دانشگاه کوچک با شبکه‌ای با چند کلاس C (به عبارتی شبکهٔ 24/) به تصویر کشیده شده است. در این شکل دو شبکهٔ اترنت وجود دارد که یکی از آنها با آدرس 192.31.65.0 به دانشکدهٔ کامپیوتر و دیگری با آدرس 192.31.63.0 به دانشکدهٔ برق تعلق دارد. این دو شبکه از طریق یک شبکهٔ حلقه (مثل FDDI) با آدرس 192.31.60.0 که نقش ستون فقرات شبکهٔ کل دانشگاه را ایفاء می‌کند، به یکدیگر متصل شده‌اند. هر ماشین متصل به اترنت دارای یک آدرس MAC یکتا است که در شکل با برچسبهای E1 تا E6 مشخص شده‌اند. هر ماشین متصل به شبکهٔ حلقه نیز آدرس FDDI خود را دارد که آنها نیز با F1 تا F3 نشان داده شده‌اند.



شکل ۵-۶۲. سه شبکهٔ بهم متصل با الگوی 24/: دو شبکهٔ اترنت و یک شبکهٔ حلقه از نوع FDDI.

حال ابتدا بررسی کنیم که کاربری بر روی ماشین میزبان ۱ چگونه بسته‌ای را برای کاربری بر روی ماشین ۲ می‌فرستد. فرض را بر آن گذاشته‌ایم که فرستنده، نام گیرندهٔ مورد نظر را می‌داند و مثلاً این نام mary@eagle.cs.uni.edu است. در اولین گام باید آدرس IP ماشین eagle.cs.uni.edu را بدست بیاورد. این جستجو از طریق «سیستم نام دامنه» (DNS) که در فصل هفتم آن را مطالعه خواهیم کرد، انجام می‌گیرد. در اینجا فرض می‌کنیم که DNS، آدرس IP ماشین ۲ را 192.31.65.5 برگردانده است.

در اینجا نرم‌افزار لایهٔ بالاتر در ماشین ۱ بسته‌ای می‌سازد و درون فیلد «آدرس مقصد» آن مقدار 192.31.65.5 را درج می‌کند و آن را جهت ارسال تحویل نرم‌افزار IP می‌دهد. نرم‌افزار IP با بررسی آدرس بدین نتیجه می‌رسد که ماشین مقصد، در شبکهٔ خودش قرار گرفته است[1] ولی محتاج روشی است تا بتواند آدرس اترنت ماشین مقصد را پیدا کند. یک راه حل آنست که یک فایل پیکربندی در جایی از سیستم ذخیره شده باشد و از طریق آن آدرس IP به آدرسهای اترنت نگاشته شود. اگرچه این روش عملی است ولی برای موسساتی که هزاران ماشین دارند به روز نگه داشتن چنین فایلهایی، کاری بسیار وقتگیر و منشاء خطاهای سهوی است.

راه حل بهتر آنست که ماشین میزبان ۱ بسته‌ای را به صورت فراگیر (Broadcast) بر روی اترنت پخش کند و از همه سؤال کند که: «آدرس 192.31.65.5 متعلق به کیست؟» این اعلام همگانی به یکایک ماشینهای شبکهٔ اترنت با آدرس 192.31.65.0 خواهد رسید. تنها ماشین ۲ به این سؤال پاسخ خواهد داد و آدرس اترنت خود یعنی E2 را اعلام خواهد کرد. بدین طریق ماشین ۱ متوجه می‌شود که آدرس 192.31.65.5 متعلق به ماشینی است که آدرس

---

۱. نرم‌افزار IP از آنجا متوجه می‌شود که ماشین مقصد در شبکهٔ محلی خودش واقع است که طبق الگوی 24/ بخش شمارهٔ شبکه خودش و شمارهٔ شبکهٔ مقصد یکسان است. (شمارهٔ شبکهٔ هردو 192.31.65.0 است.) -م

اترنت آن E2 است. پروتکلی که برای این پرسش و پاسخ بکار می‌رود **ARP** نام دارد و تقریباً تمام ماشینهای اینترنت آنرا اجرا کرده‌اند. ARP در سند RFC 826 تبیین شده است.

مزیت استفاده از ARP به جای ذخیره فایلهای پیکربندی، سادگی آنست. مدیر سیستم فقط باید برای هر ماشین، آدرس IP و الگوی زیرشبکهٔ (Subnet Mask) آنرا مشخص کند. مابقی کارها را ARP انجام می‌دهد.

پس از بدست آمدن آدرس E2 ، نرم‌افزار IP در ماشین میزبان ۱ ، یک فریم اترنت به آدرس E2 ساخته و بستهٔ IP (با آدرس 192.31.65.5) را در درون فیلد دادهٔ آن قرار داده و آن را بر روی اترنت روانه می‌کند. کارت شبکهٔ ماشین ۲ این فریم را گرفته و تشخیص می‌دهد که متعلق به خود اوست؛ آنرا پذیرفته و یک «وقفه»[1] تولید می‌کند. نرم‌افزار راه‌انداز اترنت، بستهٔ IP را از درون فریم استخراج کرده و آنرا به نرم‌افزار IP تحویل می‌دهد. نرم‌افزار IP متوجه می‌شود که این بسته به آدرس صحیحی آمده و طبعاً آن را پردازش می‌نماید.

برای آنکه ARP کارآمدتر عمل کند می‌توان بهینه‌سازیهایی انجام داد. اولین بهینه‌سازی آنست که هر گاه ARP اجرا شد و آدرسی را بدست آورد، آن را در «حافظهٔ نهان»[2] خود ذخیره نماید تا در دفعات بعدی بتواند با استفاده از این آدرس، سریعاً با ماشین مربوطه ارتباط برقرار کند. دفعهٔ بعد، ARP نگاشت آدرس IP به آدرس اترنت را در حافظهٔ نهان خود خواهد یافت و نیاز مجدد به سؤال همگانی نیست. در بسیاری از حالات، ماشین ۲ (پس از دریافت بسته) باید پاسخی پس بفرستد و در نتیجه او نیز مجبور است برای یافتن آدرس اترنت فرستندهٔ تقاضا، ARP را اجرا نماید. برای آنکه در اینجا از ارسال فراگیر بسته‌های ARP جلوگیری شود می‌توان بدین نحو عمل کرد که وقتی مثلاً ماشین ۱ برای یافتن آدرس ماشین ۲ به صورت پخش فراگیر از همه سؤال می‌کند، «نگاشت آدرس IP به آدرس اترنت» خود را نیز اعلام نماید. وقتی بسته‌های پخشی ARP به ماشین ۲ می‌رسد جفت آدرس (E1 و 192.31.65.7) وارد حافظهٔ نهان ARP می‌شود تا برای استفاده‌های آتی نیاز به سؤال نباشد. در حقیقت تمام ماشینهای متصل به اترنت می‌توانند این جفت آدرس نگاشته شده را در حافظهٔ نهان ARP خود ذخیره نمایند.[3]

روشی دیگر برای بهینه‌سازی ARP آنست که هر ماشین به محض راه‌اندازی (بوت) «نگاشت آدرس سخت‌افزاری» به آدرس IP خود را به صورت پخش فراگیر به همه اعلام نماید.[4] عموماً این کار بدین نحو انجام می‌گیرد که ماشین در حال بوت ، آدرس سخت‌افزاری معادل با IP خودش را سؤال می‌کند!! طبعاً چنین سؤالی هیچ‌پاسخی ندارد ولی نتیجهٔ جانبی آن این است که در اثر پخش این سؤال (که پاسخ خود را بهمراه دارد) بر روی کل شبکهٔ محلی، نگاشت آدرس او به صورت یک درایه (Entry) درون حافظهٔ نهان ARP ماشینهای فعال درج می‌شود. اگر پاسخ چنین سؤالی به صورت غیرمنتظره دریافت گردد، مشخص می‌شود که دو ماشین دارای آدرس IP مشابه هستند. ماشین دوم باید این موضوع را به مدیر سیستم اطلاع داده و بوت نشود.

برای آنکه جدول نگاشت آدرس در ARP بتواند تغییر کند، درایه‌های موجود در حافظهٔ نهان ARP باید پس از چند دقیقه اعتبار خود را از دست بدهند و با سؤال مجدّد، نگاشت آدرسها از نو انجام شود چراکه مثلاً اگر یک کارت شبکهٔ اترنت، خراب و با کارتی جدید عوض شود آدرس اترنت معادل با آدرس IP آن ماشین عوض می‌شود و اگر بقیهٔ ماشینها بخواهند از جدول نگاشت قدیمی خود استفاده کنند این ماشین از دسترس دیگران

---

۱. Interrupt ۲. ARP Cache

۳. بعبارت دیگر وقتی یک ماشین، آدرس سخت‌افزاری یک ماشین دیگر را سؤال می‌کند، جفت آدرس سخت‌افزاری و IP خود را نیز به همه اعلام می‌کند و از آن به بعد ماشینها نیازی به سؤال کردن ندارند. بدین نحو حافظهٔ نهان ARP سریعاً پر شده و فرآیند پرسش و پاسخ بیهوده و اضافی انجام نخواهد شد. -م

۴. واژه‌های آدرس MAC، آدرس فیزیکی، آدرس سخت‌افزاری، آدرس اترنت همگی معادلند و به آدرس تعبیه شده بر روی کارت شبکه (آدرس لایهٔ پیوند داده‌ها) اشاره دارند. -م

خارج می‌شود.

حال مجدداً به شکل ۵-۶۲ نگاهی بیندازید: هنگامی که ماشین ۱ بخواهد بسته‌ای برای ماشین ۴ (با آدرس 192.31.63.8) بفرستد با شکست مواجه می‌شود زیرا ماشین ۴ نمی‌تواند پخش فراگیر بسته‌های پرسش را دریافت کند (مسیریابها بسته‌های پخش فراگیر را به شبکه‌های دیگر هدایت نمی‌کنند). برای حل این مشکل دو راه وجود دارد: اول آنکه مسیریاب CS به گونه‌ای پیکربندی شود تا به تمام بسته‌های ARP که در مورد شبکهٔ 192.31.63.0 (یا سایر شبکه‌های محلی) آدرسی را سؤال می‌کنند، جواب بدهد و آدرس سخت‌افزاری خودش را اعلام نماید. در این حالت، ماشین ۱ در جدول خود یک درایه به صورت (E3 و 192.31.63.8) درج می‌نماید و تمام ترافیک خود برای ماشین ۴ را برای مسیریاب محلی می‌فرستد. این راه حل اصطلاحاً Proxy ARP نامیده می‌شود. (ARP وکالتی) راه حل دوم آنست ماشین ۱ فوراً تشخیص بدهد که مقصد مورد نظر او بر روی شبکه‌ای دیگر قرار گرفته و تمام ترافیک راه دور و غیرمحلی خود را به آدرس اترنت مسیریاب پیش‌فرض که در این مثال E3 است بفرستد. [تشخیص محلی یا غیرمحلی بودن آدرسهای IP به کمک الگوی زیرشبکه ـSubnet Maskـ میسّر است.] در این راه حل نیازی نیست که مسیریاب CS بداند که به چه شبکه‌هایی سرویس می‌دهد.

در هر حال آنچه که در ادامه اتفاق می‌افتد آنست که: ماشین ۱ بستهٔ IP را درون فیلد حمل داده از فریم اترنت جاسازی کرده و مقصد فریم را آدرس اترنت E3 قرار می‌دهد. وقتی مسیریاب CS این فریم اترنت را دریافت می‌کند و بستهٔ IP را از درون آن جدا کرده و آدرس IP آن را درون جدول مسیریابی خود جستجو می‌نماید و بر این اساس متوجه می‌شود که تمام بسته‌های متعلق به شبکهٔ 192.31.63.0 باید به مسیریاب با آدرس 192.31.60.7 تحویل شود و اگر از قبل آدرس سخت‌افزاری کارت FDDI متناظر با 192.31.60.7 را نداند، با پخش فراگیر یک بستهٔ ARP بر روی حلقه، آن را سؤال کرده و متوجه می‌شود که این آدرس F3 است. لذا بستهٔ IP را درون فیلد حمل داده از فریم FDDI قرار داده و با درج آدرس F3 در فریم جدید آن را بر روی حلقه قرار می‌دهد.

در مسیریاب EE، نرم‌افزار راه‌انداز کارت FDDI، بسته را از درون فریم استخراج کرده و آن را به نرم‌افزار IP تحویل می‌دهد و IP نیز به روش مشابه متوجه می‌شود که باید این بسته را به آدرس 192.31.63.8 بفرستد. اگر این آدرس IP درون حافظهٔ نهان ARP پیدا نشد، مجدداً با پخش فراگیر یک بستهٔ ARP بر روی شبکهٔ اترنت EE، آن را سؤال می‌کند و متوجه می‌شود که آدرس سخت‌افزاری مقصد E6 است، لذا یک فریم اترنت به آدرس E6 ساخته و بسته را مجدداً درون فیلد حمل دادهٔ آن قرار داده و فریم را بر روی شبکهٔ اترنت EE قرار می‌دهد. وقتی فریم اترنت به ماشین ۴ می‌رسد، بسته از درون آن استخراج شده و جهت پردازش تحویل نرم‌افزار IP می‌شود.

حرکت بسته‌ها از ماشین ۱ به یک شبکهٔ دور در WAN نیز به روشی مشابه با روش قبلی انجام می‌شود با این تفاوت که در مثال بالا مسیریاب CS از طریق جدول مسیریابی خود متوجه می‌شود که باید بسته را برای مسیریاب متصل به WAN با آدرس F2 بفرستد.

پروتکلهای BOOTP ، RARP و DHCP

پروتکل ARP مسئله پیدا کردن آدرس اترنت متناظر با یک IP مشخص را حل می‌کند. [به عبارت ساده‌تر آدرس IP یک ماشین را گرفته و آدرس سخت‌افزاری آن ماشین را پیدا کرده، بر می‌گرداند.] برخی اوقات وارون این مسئله باید حل شود، یعنی با داشتن آدرس اترنت یک ماشین، به آدرس IP متناظر با آن نیاز است. بالاخص زمانی با این مسئله مواجه هستیم که یک ایستگاه بدون دیسک سخت بخواهد از طریق شبکه بوت شود. بطور معمول چنین ماشینی «تصویر باینری سیستم عامل» خود را از یک سرویس‌دهندهٔ فایل تحویل گرفته و بارگذاری می‌کند. ولی چگونه می‌تواند آدرس IP خود را بفهمد؟

اولین راه حل ابداعی برای این مسئله آن بود که از پروتکل RARP[1] (تشریح شده در سند RFC 903) استفاده شود. این پروتکل امکان آن را فراهم آورده که ماشین تازه بوت شده آدرس اترنت خود را به صورت فراگیر بر روی شبکه پخش کند و بگوید: «آدرس اترنت ۴۸ بیتی من مثلاً 14.04.05.18.01.25 است. آیا کسی آدرس IP مرا می‌داند؟» سرویس دهندهٔ RARP این درخواست را می‌بیند و در فایلهای پیکربندی خودش، آدرس اترنت اعلام شده را جستجو می‌نماید و آدرس IP متناظر با آن را بر می‌گرداند.

استفاده از RARP بهتر از آنست که آدرس IP یک ماشین در «تصویر حافظه»[2] ارسالی جاسازی شود چراکه این امکان فراهم می‌آید که از «تصویر حافظهٔ» مشابهی برای تمام ماشینها بهره گرفته شود. اگر قرار باشد آدرس IP درون تصویر حافظهٔ ارسالی جاسازی شود، هر ایستگاه در شبکه به «تصویر» خاص خود احتیاج خواهد داشت.

اشکال RARP آنست که بیتهای فیلد آدرس مقصد را در تمام فریمهای ارسالی خود ۱ می‌گذارد تا بدین ترتیب این فریمها به صورت پخش فراگیر به سرویس دهندهٔ RARP برسند ولیکن فریمهای پخش شده بر روی شبکهٔ محلی، توسط مسیریابها به خارج از شبکه هدایت نمی‌شوند، فلذا بر روی هر شبکهٔ محلی باید یک سرویس دهندهٔ RARP وجود داشته باشد. برای حل این مشکل یک پروتکل خاص دیگر به نام BOOTP برای راه‌اندازی ایستگاههای بدون دیسک (Diskless) ابداع شده است. این پروتکل می‌تواند به غیر از آدرس IP ایستگاه بدون دیسک، اطلاعات اضافه‌تری را مثل آدرس IP سرویس دهندهٔ فایل (که تصویر اجرایی سیستم عامل را در اختیار دارد)، آدرس IP مسیریاب پیش فرض، الگوی زیرشبکه (Subnet Mask)، به ایستگاهها ارائه بدهد. برخلاف RARP ، در پروتکل BOOTP از بسته‌های UDP استفاده شده و مسیریابها این بسته‌ها را هدایت می‌نمایند. [لذا به ازای چندین شبکهٔ محلی که از طریق مسیریاب بهم متصل شده‌اند فقط به یک سرویس دهندهٔ BOOTP نیاز است.] پروتکل BOOTP در RFCهای 951 ، 1048 و 1084 تشریح شده است.

مشکل جدی پروتکل BOOTP آنست که جدول نگاشت آدرس IP به آدرس اترنت باید به صورت دستی تنظیم و پیکربندی شود. وقتی ماشین جدیدی به LAN اضافه می‌شود قادر به بوت شدن نیست مگر آنکه مسئول شبکه یک آدرس IP به آن انتساب داده و آن را در قالب: (**آدرس IP + آدرس اترنت**) به صورت دستی در فایل پیکربندی BOOTP وارد نماید. برای آنکه این مرحلهٔ اشکال‌زا حذف شود پروتکل BOOTP پیشرفته‌تر شد و با نام جدید DHCP[3] معرفی گردید. پروتکل DHCP این امکان را فراهم آورده که بتوان آدرس IP ایستگاهها را هم به صورت دستی و هم به صورت خودکار به آنها انتساب داد. این پروتکل در RFCهای ۲۱۳۱ و ۲۱۳۲ تشریح شده است و در اغلب سیستمها جایگزین RARP و BOOTP شده است.

شبیه به RARP و BOOTP ، پروتکل DHCP نیز متکی به یک سرویس دهندهٔ ویژه در شبکه است تا به ماشینهایی که تقاضای آدرس IP می‌کنند، آدرس و اطلاعات لازم را تقدیم نماید. لازم نیست که این سرویس دهنده بر روی همان LAN باشد که ماشین متقاضی آدرس بر روی آن واقع شده است و از آنجایی که ممکن است دسترسی به این سرویس دهنده از طریق پخش فراگیر بسته‌های تقاضا میسر نباشد فلذا بر روی هر LAN به یک «**عامل رلهٔ DHCP**» (DHCP Relay Agent) نیاز است. (به شکل ۵-۶۳ نگاه کنید.)

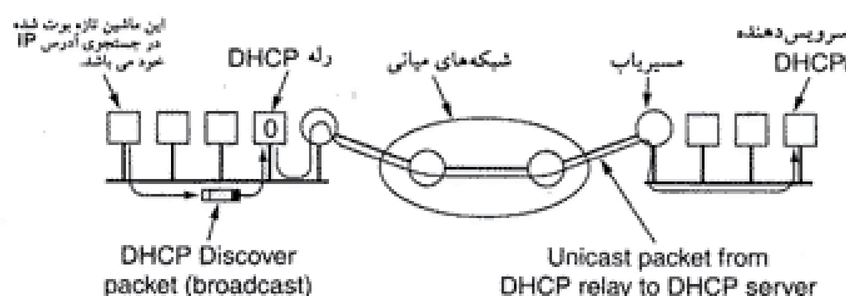
ماشینی که تازه بوت شده برای بدست آوردن آدرس IP خود، بسته‌ای به نام DHCP DISCOVER را به صورت فراگیر بر روی LAN خود منتشر می‌کند. «عامل رلهٔ DHCP» در هر LAN تمام بسته‌های پخش شده

---

۱. Reverse Address Resolution Protocol

۲. منظور از تصویر حافظه یا Memory Image قطعه کد اجرایی و باینری از هستهٔ سیستم عامل است که بلافاصله پس از بارگذاری، اجرا می‌شود. ـم

۳. Dynamic Host Configuration Protocol



شکل ۵-۶۳. عملکرد DHCP.

DHCP را می‌گیرد و وقتی متوجه شود که یک بسته از نوع DHCP DISCOVER است آن را به صورت تک‌پخشی (Unicast) برای سرویس دهندهٔ اصلی DHCP (که می‌تواند بر روی شبکه‌ای دور واقع باشد) می‌فرستد. «عامل رلهٔ DHCP» فقط به آدرس IP از سرویس دهندهٔ DHCP احتیاج دارد.

یکی از مسائلی که در خصوص انتساب خودکار آدرسها پیش می‌آید آنست که یک آدرس IP برای چه مدت زمانی در اختیار ماشین قرار بگیرد. اگر ماشینی بدون اطلاع قبلی شبکه را ترک کند و آدرس IP خود را به سرویس دهندهٔ DHCP بر نگرداند، آن آدرس برای همیشه گم می‌شود. به مرور زمان ممکن است آدرسهای زیادتری به این نحو گم شوند. برای آنکه چنین مشکلی اتفاق نیفتد، انتساب آدرسهای IP فقط برای مدت زمان محدود و ثابتی انجام می‌شود. این تکنیک «اجارهٔ IP» (IP Leasing) نام دارد. هر ماشین قبل از آنکه مهلت اجارهٔ آدرس IP او به سر برسد، باید با ارسال بسته‌ای خاص تقاضای تجدید اجاره کند. اگر ماشین موفق به ارسال این تقاضا نشد یا تقاضای تجدید پذیرفته نشود، ماشین میزبان نمی‌تواند بیش از این از آدرس IP اجاره‌ای خود استفاده نماید.[1]

۵-۶-۴ OSPF : پروتکل مسیریابی برای دروازه‌های درونی

تا اینجا بررسی پروتکلهای کنترلی اینترنت را به اتمام رسانده‌ایم. زمان آن فرا رسیده که به مورد بعدی بپردازیم: «مسیریابی در اینترنت». قبلاً اشاره کردیم که شبکهٔ اینترنت از تعداد بسیار زیادی «**سیستم خودمختار**» (Autonomous System) تشکیل شده است. هر سیستم خودمختار که اصطلاحاً AS نامیده می‌شود توسط سازمان یا نهاد خاصی پیاده و اداره می‌شود و طبیعی است که آن مؤسسه برای مسیریابی بسته‌ها در درون شبکه، از الگوریتم مورد نظر خود بهره بگیرد. به عنوان مثال شبکه‌های داخلی سه شرکت X و Y و Z، از منظر اینترنت در قالب سه AS دیده می‌شود. (البته به شرطی که به اینترنت متصل شده باشند.) از دیدگاه اینترنت، جزئیات درونی یک شبکهٔ AS قابل رویت نیست. علیرغم آنکه جزئیات درونی هر AS مستقل از دیگری است ولیکن داشتن یک استاندارد برای مسیریابی در درون یک AS پیاده‌سازی «مرز»[2] ASها را آسان می‌کند. در این بخش مسیریابی در درون یک AS و در بخش بعدی مسیریابی بین ASها را مطالعه خواهیم کرد. مسیریابی در درون یک AS اصطلاحاً «پروتکل دروازهٔ درونی»[3] و الگوریتم مسیریابی بین ASها «پروتکل دروازهٔ خارجی»[4] نامیده می‌شود.

---

۱. در DHCP مسئول شبکه تعدادی آدرس IP یا محدوده‌ای از فضای آدرس IP مورد نظر خود را مشخص می‌کند تا این سرویس دهنده آنها را برحسب نیاز به ماشینهای شبکه اجاره بدهد. به این آدرسها اصطلاحاً IP Pool گفته می‌شود. م-

۲. منظور از «مرز» نقطه‌ای است که ASها به یکدیگر متصل می‌شوند و مسیریابهایی که این اتصال را برقرار می‌کنند مسیریابهای مرزی (Border Gateway) نامیده می‌شوند. این مسیریابها هم با درون AS و هم با دیگر ASها در ارتباط فعّال هستند؛ یعنی هم مسیرهای بهینهٔ داخلی و هم مسیرهای بهینهٔ بین ASها را می‌دانند. -م

۳. Interior Gateway Protocol

۴. Exterior Gateway Protocol

اولین پروتکل «دروازهٔ درونی اینترنت»، یک پروتکل بردار فاصله با نام RIP بود که از الگوریتم «بلمن-فورد» بهره می‌گرفت. [بخش ۵-۲-۴] اگرچه این پروتکل در سیستمهای کوچک به خوبی کار می‌کند ولی با بزرگ شدن ASها، کارآیی خود را از دست می‌دهد. این پروتکل همچنین از مشکل «شمارش تا بی‌نهایت» رنج می‌برد و «همگرایی» کندی دارد. به همین دلیل در ماه می ۱۹۷۹ یک پروتکل مبتنی بر «حالت لینک» (Link State) جایگزین آن شد. در ۱۹۸۸، IETF کار را بر روی یک پروتکل جدید آغاز کرد. این پروتکل OSPF (Open Shortest Path First) نام گرفت و در سال ۱۹۹۰ استاندارد شد. امروزه اغلب تولیدکنندگان مسیریاب از آن حمایت می‌کنند و تقریباً به مهمترین پروتکل مسیریابی دروازه‌های درونی تبدیل شده است. در زیر شمائی از عملکرد OSPF ارائه می‌دهیم. برای آگاهی دقیقتر از کل قضیه به سند RFC 2328 مراجعه نمایید.

گروه طراح این پروتکل با داشتن تجربهٔ طولانی از دیگر پروتکلهای مسیریابی، تصمیم گرفتند با تدوین فهرست بالا بلندی از نیازها و انتظارات، آنرا از نو طراحی کنند. نخستین نیاز آن بود که به صورت «باز» (Open) طراحی شود بدین معنا که امتیاز راه‌حلها و روشهای خاصّ آن برای موسسهٔ خاصّی ثبت نشود و همچنین به بستر سخت‌افزاری یا نرم‌افزاری ویژه وابسته نباشد. حرف 'O' در نام OSPF به همین مضمون است.

نیاز دوم آن بود که پروتکل جدید بتواند برای تعیین مسیر بهینه، از معیارهای گوناگون هزینه مثل معیار «فاصلهٔ فیزیکی»، «تاخیر» و نظایر آن پشتیبانی کند. نیاز سوم آن بود که الگوریتم پویا باشد و هر گونه تغییر در توپولوژی زیرشبکه را به سرعت و به صورت خودکار تشخیص داده و خود را با آن تطبیق بدهد.

نیاز چهارم آنکه OSPF می‌بایست بسته‌ها را برحسب نوع خدمات درخواستی، مسیریابی و هدایت نماید. این پروتکل باید قادر می‌بود که ترافیک داده‌های بی‌درنگ را نسبت به ترافیک داده‌های معمولی به روش متفاوتی مسیریابی نماید. پروتکل IP در هر بسته یک فیلد به نام Type of Service تعریف کرده بود که هیچ پروتکل مسیریابی از آن حمایت نمی‌کرد. اگرچه این فیلد در OSPF گنجانده شد ولیکن باز هم کسی از آن استقبال نکرد و عاقبت حذف گردید.

نیاز پنجم (که با موارد فوق مرتبط است) آن بود که پروتکل جدید مکانیزم **«موازنهٔ بار»** (Load Balancing) را اعمال کند و بار را بر روی چندین خط تقسیم نماید. اغلب پروتکلها تمام بسته‌ها را بر روی بهترین مسیر ارسال می‌کنند و از مسیری که از لحاظ بهینگی در رتبهٔ دو قرار می‌گیرد هیچ استفاده‌ای نمی‌شود؛ در بسیاری از حالات تقسیم بار بر روی چندین خط، کارآیی بهتری دارد.

ششم آنکه نیاز بود از مسیریابی سلسله‌مراتبی پشتیبانی شود. تا سال ۱۹۸۸ اینترنت آنقدر بزرگ شده بود که نمی‌شد انتظار داشت یک مسیریاب بتواند توپولوژی کل آن را بداند. پروتکل جدید می‌بایست به گونه‌ای طراحی می‌شد که هیچ مسیریابی نیاز به دانستن توپولوژی کل شبکه نداشته باشد.

هفتم آنکه به سطحی از امنیت نیاز بود تا یک دانشجوی بازیگوش نتواند با ارسال اطلاعات جعلی و نادرست در خصوص مسیرها، مسیریاب را به اشتباه بیندازد. در آخر آنکه به تمهیداتی نیاز بود تا مسیریابهایی که با مکانیزم «ایجاد تونل» (Tunneling) و از طریق اینترنت بهم متصل می‌شوند را به درستی مدیریت نماید.

OSPF از سه نوع شبکه و خطوط اتصال پشتیبانی می‌کند:

۱. خطوط نقطه به نقطه بین دو مسیریاب
۲. شبکه‌های با دسترسی چندگانه که کانال آنها از نوع پخش فراگیر (Broadcast) است. (مثل اغلب شبکه‌های LAN)
۳. شبکه‌های با دسترسی چندگانه بدون آنکه کانال آنها از نوع پخش فراگیر باشد. (مثل اغلب شبکه‌های سوئیچ بسته در WAN)

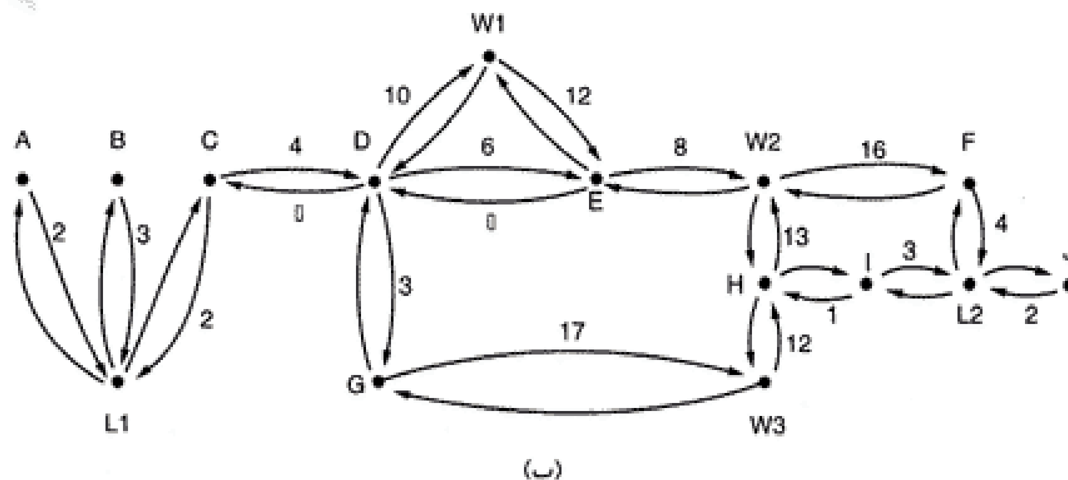
شبکهٔ با دسترسی چندگانه (Multiaccess Network)، شبکه‌ای است که می‌تواند چندین مسیریاب داشته باشد و هر یک از مسیریابها می‌توانند مستقیماً با یکدیگر در ارتباط باشند. تمام شبکه‌های LAN و WAN دارای این ویژگی هستند. شکل ۵-۶۴-الف یک AS را نشان می‌دهد که در آن هر سه نوع شبکه وجود دارد. دقت کنید که از دیدگاه OSPF ، ماشینهای میزبان شبکه عموماً نقش خاصّی را ایفاء نمی‌کنند.

OSPF بدین نحو عمل می‌کند که مجموعهٔ شبکه‌ها، مسیریابها و خطوط ارتباطی را در قالب یک «گراف جهت‌دار» (Directed Graph) مدل می‌کند و به هر کمان در گراف (Arc) یک مقدار «هزینه» (مثل تأخیر، فاصله یا امثال آن) انتساب می‌دهد. سپس براساس وزن هر یک از کمانها، مسیر بهینه را پیدا می‌کند. یک خط ارتباطی سریال بین دو مسیریاب، در گراف با یک جفت کمان (Arc) نشان داده می‌شود (یک کمان به ازای هر جهت) و وزنهای هر کمان می‌تواند با دیگری متفاوت باشد. یک شبکه با دسترسی چندگانه (LAN) ، با یک گره به ازای خود شبکه و یک گره به ازای هر مسیریاب مدل می‌شود. وزن کمانی که از گره شبکه به یک مسیریاب وارد می‌شود، صفر در نظر گرفته شده و از گراف حذف می‌گردد.

شکل ۵-۶۴-ب ، نمایش گراف متناظر با شبکهٔ ۵-۶۴-الف را نشان می‌دهد. وزنها متقارن هستند مگر آنهایی که به صراحت مشخص شده‌اند. آنچه که OSPF انجام می‌دهد مدل کردن شبکه در قالب گرافی شبیه به این مثال و سپس محاسبهٔ مسیرهای بهینه از هر مسیریاب به هر مسیریاب دیگر است.

WAN 1
A B C D E WAN 2 F
LAN 1
H I J
LAN 2
G
WAN 3

(الف)



(ب)

شکل ۵-۶۴. (الف) یک سیستم خودمختار (ب) نمایش گراف از شکل الف.

بسیاری از ASها در اینترنت خودشان بسیار عظیم هستند و مدیریت آنها ساده نیست. OSPF این امکان را فراهم آورده که بتوان چنین شبکه‌هایی را به تعدادی **«ناحیهٔ شماره‌گذاری شده»** (Numbered AREA) تقسیم کرد. هر ناحیه خود یک شبکه یا مجموعه‌ای از شبکه‌های بهم پیوستهٔ مجاور است. نواحی با یکدیگر همپوشانی ندارند[1] ولی لازم نیست که نواحی تعریف شده کل شبکهٔ AS را پوشش بدهد و ممکن است برخی از مسیریابها در هیچ ناحیه‌ای قرار نگیرند. یک ناحیه، شکل کلی و عمومی یک زیرشبکهٔ مستقل است و در خارج از ناحیه، توپولوژی و جزئیات درونی آن مشهود نیست.

در هر شبکهٔ خودمختار (AS) ناحیه‌ای به نام **«ستون فقرات»** وجود دارد که ناحیهٔ صفر نامیده می‌شود. تمام نواحی به ستون فقرات متصل می‌شوند (به صورت مستقیم یا توسط ایجاد تونل) لذا براحتی می‌توان به کمک ستون‌فقرات از هر ناحیه در شبکهٔ AS به ناحیهٔ دیگر رفت. یک «تونل» نیز در گراف توسط یک کمان مشخص می‌شود که دارای هزینه است. هر مسیریاب که به دو یا چند ناحیه متصل باشد (یعنی مسیریابهای مشترک بین دو یا چند ناحیه) جزیی از ستون فقرات شبکه محسوب می‌شود. همانند بقیهٔ نواحی، توپولوژی ناحیهٔ ستون فقرات نیز در خارج از آن مشهود نیست. (بعبارتی مسیریابهای درون دیگر نواحی از توپولوژی ستون‌فقرات چیزی نمی‌دانند.)

درون یک ناحیه، هر یک از مسیریابها نسخهٔ مشابهی از پایگاه اطلاعاتی در خصوص مسیرها و هزینه‌ها در اختیار دارند و الگوریتم محاسبهٔ کوتاهترین مسیر آنها یکسان است. هر مسیریاب وظیفه دارد که کوتاهترین مسیرها از خودش به تمام مسیریابهای دیگر ناحیه را محاسبه نماید. از جمله هر مسیریاب باید کوتاهترین مسیر از خود تا یک مسیریاب واقع بر روی ستون فقرات را پیدا کند. یک مسیریاب که در مرز دو ناحیه واقع است باید پایگاه اطلاعاتی هر دو ناحیه را در اختیار داشته باشد و الگوریتم کوتاهترین مسیر را بطور جداگانه بر روی آنها اجرا کند. [تا تمام مسیرهای بهینه از خودش تا بقیهٔ مسیریابها در هر دو ناحیه بدست بیاید.]

در حین عملیات طبیعی، احتمالاً به سه نوع مسیر نیاز است: (۱) مسیرهای درون ناحیه[2] (Intra-Area) (۲) مسیرهای بیرون ناحیه (Inter-Area) (۳) مسیرهای بین AS (Inter-AS).

مسیرهای درون ناحیه ساده‌ترین نوع مسیر هستند چراکه مسیریاب مبداء، از قبل و به دقت کوتاهترین مسیر رسیدن به مسیریاب مقصد را می‌داند. مسیریابی بین نواحی (Inter-Area)، همیشه در سه مرحله انجام می‌گیرد: حرکت از مبداء تا ستون فقرات، سپس حرکت از ستون فقرات به سوی ناحیهٔ مقصد و نهایتاً حرکت در درون ناحیه به سمت مقصد. این الگوریتم ایجاب می‌کند که در پروتکل OSPF یک توپولوژی **«ستاره»** برای مسیریابها قائل شویم: ستون فقرات در مرکز قرار می‌گیرد و بقیهٔ نواحی از آن منشعب می‌شوند. در OSPF بسته‌ها به همان نحوی که از مبداء تولید شده‌اند به سوی مقصد مسیریابی و هدایت می‌شوند یعنی بسته‌ها در درون بستهٔ دیگری «جاسازی» (Encapsulate) یا «تونل» (Tunnel) نمی‌شوند، مگر آنکه بسته مجبور باشد برای رسیدن از یک ناحیه به ستون‌فقرات، از مسیری حرکت کند که از طریق تونل ایجاد شده است.[3] شکل ۵-۶۵ بخشی از اینترنت را با چهار AS و تعدادی ناحیه نشان می‌دهد.

OSPF چهار کلاس مسیریاب را به رسمیت می‌شناسد:

۱. **مسیریابهای درونی (که کاملاً در داخل یک ناحیه قرار می‌گیرند.)**
۲. **مسیریابهای واقع بر مرز دو ناحیه (که دو یا چند ناحیه را به هم متصل می‌کنند.)**

---

۱. یعنی هر مسیریاب صرفاً به یک ناحیه متعلق است. -م
۲. یعنی مسیرهایی که از یک مسیریاب در درون ناحیه شروع و به یک مسیریاب در همان ناحیه ختم می‌شود. -م
۳. مفهوم تونل را در بخش ۵-۵-۵ مطالعه نمائید.

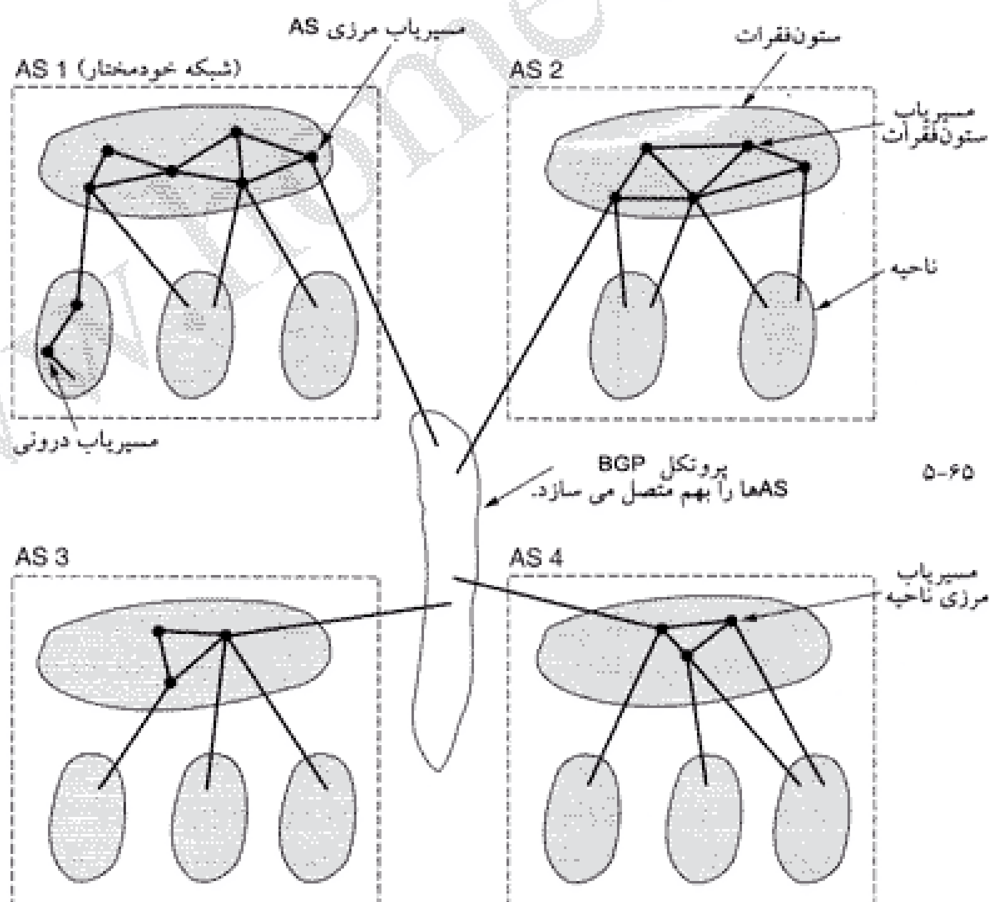
۳. مسیریابهای ستون‌فقرات (که بر روی ستون‌فقرات قرار می‌گیرند.)

۴. مسیریابهای مرزی AS که می‌توانند با مسیریابهای AS دیگر محاوره کنند.

البته امکان آن وجود دارد که مسیریابها در دو یا چند کلاس قرار بگیرند. به عنوان مثال تمام مسیریابهای مرزی، به صورت خودکار جزیی از ستون فقرات نیز هستند. مضاف بر این، یک مسیریاب که بر روی ستون فقرات واقع است ولی در هیچ ناحیه‌ای قرار نگرفته، خودش یک مسیریاب داخلی محسوب می‌شود. در شکل ۵-۶۵ نمونه‌ای از تمام کلاسهای مسیریاب، دیده می‌شود.

وقتی یک مسیریاب بوت می‌شود، ابتدا بر روی تمام خطوط نقطه‌به‌نقطه و کانالهای اشتراکی (مثل کانالهای LAN که چند مسیریاب از طریق آن بهم متصل شده‌اند)، پیامی به نام «**پیام سلام**» (Hello Message) می‌فرستد. در خطوط WAN [مثل خطوط ISDN] احتمالاً قبل از هر گونه مبادلهٔ پیام به اطلاعات پیکربندی خاص نیاز است تا هویت تماس‌گیرنده مشخص شود.[1] هر مسیریاب با دریافت پاسخ سلام، مسیریابهای همسایهٔ خود را شناسایی می‌کند. مسیریابهایی که به یک LAN واحد متصلند، همگی همسایهٔ یکدیگر محسوب می‌شوند.

عملکرد OSPF مبتنی بر مبادلهٔ اطلاعات با «مسیریابهای مجاور» (Adjacent) است. مفهوم مسیریابهای مجاور با مفهوم «مسیریابهای همسایه» (Neighbor) فرق می‌کند؛ اینکه هر مسیریاب متصل به LAN بتواند با مسیریاب دیگر محاوره و مبادلهٔ پیام کند کافی و کارآمد نیست. برای اجتناب از این وضعیت، از بین کل مسیریابهای



شکل ۵-۶۵. ارتباط بین ASها، ستون‌فقرات و نواحی در OSPF.

۱. مثل تنظیمات خط و شمارهٔ PIN

واقع بر LAN، یک مسیریاب به نام «**مسیریاب برگزیده**» (Designated Router) انتخاب می‌گردد. این مسیریاب، «مسیریاب مجاور» بقیهٔ مسیریابهای واقع بر LAN محسوب می‌شود و با آنها به مبادلهٔ اطلاعات می‌پردازد. برای «مسیریاب برگزیده» یک مسیریاب پشتیبان نیز در نظر گرفته شده که آن نیز همیشه اطلاعات به‌روز از وضعیت مسیرها در اختیار دارد تا در صورت از کار افتادن «مسیریاب برگزیده»، به سرعت جایگزین آن شود.

در حین عملکرد طبیعی، هر مسیریاب بطور متناوب پیامهای LINK STATE UPDATE خود را به صورت سیل‌آسا(Flooding) برای تمام مسیریابهای مجاور خود می‌فرستد. [این پیامها حاوی اطلاعاتی در خصوص هویت همسایه‌ها، لینکها و هزینهٔ آنهاست.] بدین ترتیب، با این پیامها حالت هر لینک (Link State) و هزینهٔ آن، برای درج در «**پایگاه اطلاعات توپولوژیکی**» به دست می‌آید. وصول پیامهایی که به صورت سیل‌آسا ارسال می‌شوند، تصدیق خواهد شد تا از دریافت آنها اطمینان حاصل شود.[1] هر پیام یک شمارهٔ ترتیب دارد تا مسیریاب بفهمد آیا پیام LINK STATE UPDATE قدیمی یا جدید است. ارسال این پیامها به صورت سیل‌آسا زمانی آغاز می‌شود که خطی از کار بیفتد یا مجدداً فعال شود یا هزینهٔ آنها تغییر کند. البته حتی اگر چنین اتفاقی نیفتد، این پیامها بطور متناوب و هر از چند ده ثانیه ارسال خواهند شد.

پیام DATABASE DESCRIPTION تمام شماره‌های ترتیب از درایه‌های جدول حالت لینک را که اخیراً در پایگاه اطلاعاتی فرستندهٔ آن ذخیره شده، اعلام می‌کند. هر یک از گیرندگان این پیام، شماره‌های اعلام شده را با شماره‌های ترتیب درایه‌های خودش مقایسه می‌کند تا بفهمد چه کسی جدیدترین مقادیر را در اختیار دارد. از این پیامها زمانی استفاده می‌شود که خطی فعال شود.

هر یک از طرفین یک لینک می‌توانند با استفاده از پیام LINK STATE REQUEST، «اطلاعات حالت پیوند» یکدیگر را مطالبه نمایند. نتیجهٔ این الگوریتم آنست که هر جفت مسیریاب مجاور، آخرین اطلاعات به‌روز شدهٔ یکدیگر را بررسی کرده و بدین نحو اطلاعات جدید در کل ناحیه پخش می‌شود. تمام این پیامها به کمک بسته‌های معمولی IP ارسال می‌شود. فهرست پنج پیام فوق در جدول ۵-۶۶ خلاصه شده است.

نوع پیام	توصیف عملکرد
Hello	از این پیام برای شناسایی همسایه‌ها استفاده می‌شود.
Link state update	هزینه فرستنده پیام تا همسایه‌هایش را معین می‌کند.
Link state ack	دریافت بسته Link State Update را تایید می‌کند.
Database description	مسیریاب با این پیام فهرست درایه‌های بهنگام‌سازی خود را اعلام می‌کند.
Link state request	از شریک خود اطلاعاتی را درخواست می‌کند.

شکل ۵-۶۶. پنج نوع از پیامهای OSPF.

حال می‌توان مجموعهٔ عوامل فوق را بدین نحو جمع‌بندی کرد: به کمک روش ارسال سیل‌آسا، هر مسیریاب به تمام اعضاء ناحیهٔ خود، از همسایه‌هایش و هزینهٔ رسیدن به آنها خبر می‌دهد. این اطلاعات امکان آن را فراهم می‌آورد تا یکایک مسیریابها بتوانند گراف ناحیهٔ خود را تشکیل داده و کوتاهترین مسیرها را محاسبه نمایند. ستون فقرات نیز همین کار را می‌کند. [چراکه ستون فقرات خود یک ناحیهٔ مستقل است.] مضاف بر این، مسیریابهای واقع بر روی ستون فقرات، اطلاعات ارسالی از مسیریابهای مرزی هر ناحیه را هم می‌پذیرند تا بتوانند کوتاهترین مسیرها از ستون فقرات تا دیگر مسیریابها را محاسبه نمایند. این اطلاعات مجدداً به مسیریابهای مرزی هر ناحیه

---

۱. یعنی به ازای دریافت هر پیام از این نوع یک پیام ACK بر می‌گردد.

بازگردانده می‌شود تا آنها نیز در درون ناحیهٔ خودشان اعلام نمایند. به کمک این اطلاعات، یک مسیریاب که می‌خواهد بسته‌ای را به خارج از ناحیهٔ خود بفرستد، بهترین مسیریاب مرزی متصل به ستون فقرات را به عنوان دروازهٔ خروج بسته برمی‌گزیند.

۵-۶-۵ BGP[1] : پروتکل مسیریابی برای دروازهٔ خارجی

در درون یک AS واحد، پروتکل مسیریابی OSPF، بهترین گزینه است (اگرچه OSPF، تنها پروتکل مورد استفاده و رایج به حساب نمی‌آید). بین ASها از پروتکل متفاوتی به نام BGP استفاده می‌شود. به دلیل آنکه اهداف پروتکل مسیریابی درونی با پروتکل مسیریابی خارجی فرق می‌کند به پروتکل متفاوتی برای مسیریابی بین ASها نیاز است. پروتکل مسیریابی درونی باید به سریعترین وجه ممکن بسته‌ها را از مبداء به مقصد برساند و اهمیتی به سیاستهای جانبی نمی‌دهد.

پروتکلهای مسیریابی خارجی بایستی در حد وسیعی ملاحظات سیاسی را مدنظر قرار بدهند. (Metz, 2001) برای مثال ممکن است یک شبکهٔ AS بخواهد که بسته‌ها را به هر سایت اینترنت بفرستد یا بسته‌ها را از هر سایت اینترنت دریافت کند ولی تمایلی به حمل بسته‌هایی که از یک AS خارجی تولید شده و به یک AS خارجی دیگر می‌رود نداشته باشد، حتی اگر AS او بر روی کوتاهترین مسیر بین دو AS خارجی قرار گرفته باشد. (یعنی با این استدلال که: «مشکل آنها به ما ربطی ندارد!» بسته‌های دیگران را ترانزیت نکند.) از طرف دیگر ممکن است بخواهد در ازای دریافت هزینهٔ خدمات، ترافیک همسایه‌های خود و ASهای خاصّی را مسیریابی و هدایت کند. مثلاً شرکتهای مخابرات تلفن ممکن است مایل باشند به عنوان حامل مشتریان خود عمل کنند ولی نه برای دیگران (و بدون اخذ وجه)! پروتکلهای مسیریابی خارجی بطور عامّ و پروتکل BGP بطور خاصّ این امکان را فراهم کرده‌اند که بتوان سیاستهای گوناگونی را بر روی ترافیک بین ASها اعمال کرد.

سیاستهای کلی حول مسائل سیاسی، امنیتی یا ملاحظات اقتصادی دور می‌زند. چند مثال از ملاحظات مسیریابی عبارتند:

۱. عدم اجازهٔ عبور ترافیک داده‌ها از میان ASهای خاصّ
۲. مسیری که از پنتاگون شروع می‌شود هرگز از عراق عبور نکند!
۳. مسیر رسیدن از بریتیش‌کلمبیا به اونتاریو از ایالات متحده نگذرد!
۴. فقط زمانی از مسیر آلبانی عبور شود که هیچ راه دیگری به مقصد وجود نداشته باشد!
۵. ترافیک داده‌هایی که از IBM شروع شده یا بدان ختم می‌شود از مایکروسافت عبور نکند!!

عموماً سیاستهای مسیریابی، به صورت دستی بر روی مسیریابهای BGP تنظیم و پیکربندی می‌شود (یا در قالب نوعی «اسکریپت» بر روی مسیریاب اجرا می‌گردد). البته این تنظیمات جزیی از پروتکل بحساب نمی‌آید.

از دیدگاه یک مسیریاب BGP، کل جهان از چندین AS و خطوط ارتباطی بین آنها تشکیل شده است. دو AS وقتی متصل تلقی می‌شوند که بین مسیریابهای مرزی هر یک از آنها حداقل یک خط وجود داشته باشد. با توجه به تأکید BGP بر حمل (یا بعبارتی ترانزیت) ترافیک، هر شبکهٔ AS در یکی از سه ردهٔ ذیل قرار می‌گیرد: (۱) رده اول **«شبکه‌های پایانی»** (Stub Network) نامیده می‌شوند؛ این گونه از شبکه‌ها فقط و فقط یک اتصال با گراف BGP دارند و نمی‌توانند ترافیک داده‌ها را از خود عبور بدهند چراکه در طرف دیگر آنها شبکه‌ای وجود ندارد.[2] (۲) ردهٔ بعدی، **«شبکه‌های چنداتصالی»** (Multiconnected Networks) هستند. این شبکه‌ها می‌توانند حمل ترافیک

---

۱. Border Gateway Protocol

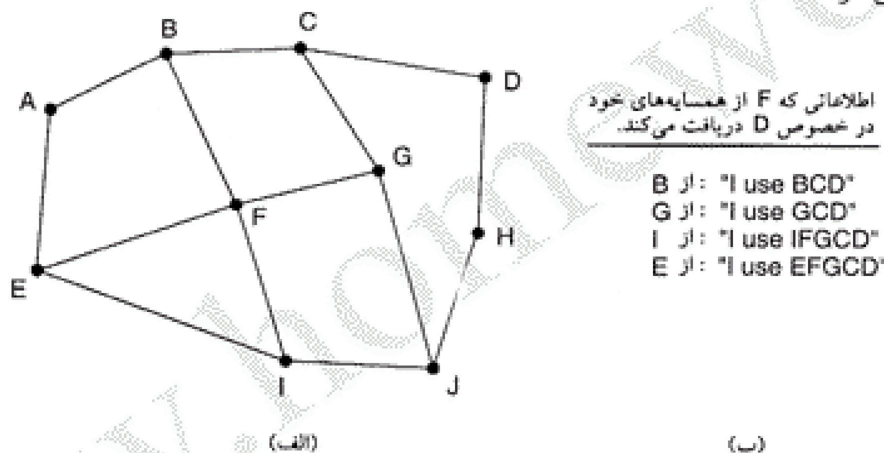
۲. شبکه‌های پایانی را به مثابهٔ یک کوچهٔ بن‌بست فرض کنید. -م

بسته‌های دیگران را بر عهده بگیرند مگر آنکه عمداً از این کار امتناع کنند. (۳) در ردهٔ آخر، **«شبکه‌های ترانزیت»** (Transit Network) قرار می‌گیرند که در نقش ستون فقرات، تمایل دارند بسته‌های شبکهٔ دیگران را (طبق برخی از محدودیتها و احتمالاً دریافت هزینه) حمل (ترانزیت) کنند.

یک جفت مسیریاب BGP از طریق برقراری یک اتصال TCP ، با یکدیگر مبادلهٔ اطلاعات می‌نمایند. این مکانیزم، امکان مبادلهٔ مطمئن و بدون خطای داده‌ها را فراهم کرده و جزئیات درونی شبکه را پنهان نگاه می‌دارد.

پروتکل BGP ذاتاً مبتنی بر الگوریتم بردار فاصله است ولی از بسیاری جهات با RIP فرق دارد؛ تفاوت بنیانی در اینجاست که به جای آنکه فقط هزینه و خط رسیدن به یک مقصد در شبکه محاسبه و نگهداری شود، مسیریاب BGP کل مسیر رسیدن به هر مقصد را نگه می‌دارد. همچنین به جای آنکه به همسایه‌های خود در خصوص هزینهٔ رسیدن به هر مقصد در شبکه ، خبر بدهد کلّ مسیرهای واقعی را اعلام می‌کند.

به عنوان مثال مسیریابهای BGP در شکل ۵-۶۷-الف و به ویژه جدول مسیریابی F را در نظر بگیرید. فرض کنید که این مسیریاب برای رسیدن به D از مسیر FGCD استفاده می‌کند. وقتی همسایه‌ها اطلاعات مسیریابی خود را به او می‌دهند مسیرهای کامل را مشابه فهرست ۵-۶۷-ب اعلام می‌نمایند. (برای سادگی فقط بخشی از مسیرها که به D ختم می‌شوند نشان داده شده است.)



شکل ۵-۶۷. (الف) مجموعه‌ای از مسیریابهای BGP. (ب) اطلاعات ارسالی برای F.

پس از آنکه F تمام مسیرها را از همسایه‌های خود دریافت کرد، آنها را بررسی می‌کند تا ببیند کدامیک از آنها بهترین است. مثلاً F برای تعیین بهترین مسیر رسیدن به D ، فوراً مسیرهای اعلام شده توسط I و E را حذف می‌کند چراکه این مسیرها خودشان از F می‌گذرند. انتخاب نهایی از بین مسیرهای اعلام شدهٔ B و G است. هر مسیریاب BGP دارای یک ماجول خاص است که مسیرهای رسیدن به یک مقصد خاص را بررسی کرده و به آنها «نمره» می‌دهد و در نهایت برای هر مسیر عددی را به عنوان «معیار فاصله» (Distance) بر می‌گرداند. هر مسیری که ملاحظات و محدودیتهای اتخاذ شده را نقض کرده باشد، نمرهٔ بی‌نهایت می‌گیرد. پس از این مرحله، مسیریاب مسیری با کمترین هزینه را می‌پذیرد. تابع نمره‌دهی[1] به مسیرها بخشی از پروتکل BGP بحساب نمی‌آید و می‌تواند به دلخواه مدیر سیستم انتخاب و تعریف شود.

پروتکل BGP به سادگی مشکل «شمارش تا بی‌نهایت» (که الگوریتمهای بردار فاصله را آزار می‌دهد) حل کرده است.[2] به عنوان مثال فرض کنید G از کار بیفتد یا خط FG قطع شود. لذا F فقط از سه همسایهٔ باقیماندهٔ خود (یعنی B و I و E) اطلاعات مسیر می‌گیرد. مسیرهائی که همسایه‌های او برای رسیدن به D اعلام می‌کنند عبارتند از :

۱. Scoring Function
۲. زیرا مسیرهای کامل به همسایه‌ها اعلام می‌شود. بخش ۵-۲-۴ را ببینید. -م

IFGCD ، BCD و EFGCD . مسیریاب F بلافاصله متوجه می‌شود که مسیرهای دوم و سوم بی‌فایده هستند چراکه از خود F می‌گذرند، لذا FBCD به عنوان مسیر جدید انتخاب می‌شود. الگوریتمهای دیگر «بردار فاصله» بدان دلیل در انتخاب مسیر به اشتباه می‌افتند که همسایه‌ها نمی‌توانند اعلام کنند کدامیک از مسیرهای رسیدن به مقصد، مسیر مستقلی است. (یعنی از خود همسایه نمی‌گذرد). پروتکل BGP در اسناد RFC 1771 تا RFC 1774 تشریح شده است.

۶-۶-۵ ارسال چندپخشی در اینترنت (Internet Multicasting)

عملکرد طبیعی پروتکل IP ، مبادلهٔ بسته‌های داده بین **یک** گیرنده و **یک** فرستنده است؛ با این وجود در برخی از کاربردها این قابلیت که یک پروسه بتواند بطور همزمان برای تعداد بسیار زیادی گیرنده، ارسال داشته باشد، کارساز و مفید است. مثالهایی از این قبیل عبارتست از: به هنگام‌سازی همزمان نسخه‌های توزیع شده یا تکراری پایگاههای اطلاعاتی، اعلام همزمان قیمت سهام به متولیان و واسطه‌ها، مدیریت و اجرای کنفرانسهای دیجیتال و گردهم‌آیی تلفنی.

IP با استفاده از آدرسهای کلاس D از فرآیند چندپخشی (Multicasting) پشتیبانی می‌نماید. هر آدرس کلاس D، هویت یک «گروه» از ماشینها را مشخص می‌کند. برای مشخص کردن هر گروه در کلاس D ، ۲۸ بیت در اختیار است، لذا بطور همزمان می‌توان بیش از ۲۵۰ میلیون گروه تعریف کرد. هر گاه یک پروسه، بسته‌ای را به یک آدرس کلاس D ارسال کند، برای رساندن آن بسته به یکایک اعضای گروه، مسیریابها حداکثر تلاش ممکن را بعمل می‌آورند ولیکن هیچ تضمینی داده نمی‌شود و احتمال دارد برخی از اعضا بسته را دریافت نکنند!

در IP از دو نوع آدرس گروهی حمایت می‌شود: **آدرسهای دائم** (Permanent) و **آدرسهای موقت** (Temporary). «گروه دائم» همیشه وجود دارد و نیازی به تنظیم و هماهنگی قبلی نیست. هر گروه دائم دارای آدرس ثابت و بلاتغییر است. چند نمونه از آدرس گروههای ثابت عبارتند از:

گروه 224.0.0.1 : تمام سیستمهای متصل به یک LAN
گروه 224.0.0.2 : تمام مسیریابهای متصل به یک LAN
گروه 224.0.0.5 : تمام مسیریابهای OSPF متصل به یک LAN
گروه 224.0.0.6 : تمام مسیریابهای «برگزیدهٔ OSPF» متصل به یک LAN

گروههای موقتی باید قبل از استفاده، ایجاد شوند. یک پروسه می‌تواند از ماشین خود درخواست کند که به یک گروه خاص بپیوندد؛ همچنین می‌تواند از آن بخواهد که گروه را ترک کند. وقتی که آخرین پروسه در یک ماشین، گروهی را ترک کند، ماشین مربوطه از آن گروه خارج خواهد شد. هر ماشین می‌داند که پروسه‌های او عضو چه گروههایی هستند.

ارسال چندپخشی توسط مسیریابهای خاصی پیاده‌سازی و پشتیبانی می‌شود و طبعاً ممکن است در کنار مسیریابهای استاندارد، اینگونه مسیریابها وجود نداشته باشد، لذا توانایی ارسال چندپخشی به نوع مسیریابهای شبکه و پیکربندی آنها برمی‌گردد. مسیریابهای چندپخشی تقریباً در هر دقیقه یکبار، یک پیام سخت‌افزاری (یعنی یک فریم لایهٔ پیوند داده‌ها) را به صورت چندپخشی برای تمام ماشینهای متصل به LAN خودش (یعنی به آدرس 224.0.0.1) می‌فرستد و از آنها می‌خواهد که اعلام کنند پروسه‌هایشان در چه گروههایی عضو هستند. در پاسخ به این سؤال، یکایک ماشینها، آدرسهای کلاس D گروههای مورد نظر خود را بر می‌گردانند.

این بسته‌های پرسش و پاسخ مبتنی بر پروتکلی به نام IGMP[1] است که تا حدودی به ICMP شبیه است. در

1. Internet Group Management Protocol

این پروتکل فقط دو نوع بسته تعریف شده است: بستهٔ پرسش و بستهٔ پاسخ. هر یک از این بسته‌ها دارای قالبی ساده و ثابت هستند: در اولین کلمه از فیلد داده (PayLoad) برخی اطلاعات کنترلی قرار می‌گیرد و در کلمهٔ بعدی یک آدرس کلاس D درج می‌شود. [کلمات چهار بایتی هستند.] این پروتکل در سند RFC 1112 تشریح شده است.

مسیریابی چندپخشی در اینترنت به کمک «درختهای پوشا» (Spanning Tree) انجام می‌شود. هر مسیریاب که از فرآیند چندپخشی حمایت می‌کند، به کمک «پروتکل اصلاح شدهٔ بردار فاصله» با همسایه‌های خود اطلاعاتی را مبادله می‌کند تا هر کدام بتوانند به ازای هر گروه یک «درخت پوشا» تشکیل بدهند. (به نحوی که تمام اعضای هر گروه را در بر بگیرد.) به منظور آنکه درخت به نحوی سازماندهی و پیرایش شود تا مسیریابها یا شبکه‌های غیر عضو در گروه از ساختار درخت حذف گردند، بهینه‌سازیهایی صورت گرفته است. این پروتکل از مکانیزم «ایجاد تونل» استفادهٔ زیادی می‌کند تا بدین نحو برای گره‌هایی که در درخت پوشا قرار ندارند سردرگمی و اشکال ایجاد نشود.

۵-۶-۷ IP متحرک (Mobile IP)

بسیاری از کاربران اینترنت از کامپیوترهای کیفی قابل حمل استفاده می‌کنند و طبعاً علاقمندند وقتی به یک محل جدید نقل مکان می‌نمایند یا حتی در طول راه، اتصالشان با اینترنت برقرار بماند. متأسفانه، سیستم آدرس‌دهی IP به گونه‌ای طراحی شده که کار کردن با آن دور از خانه، فقط در گفتار ساده است تا در عمل! در این بخش این مشکل و راه حل آن را بررسی می‌کنیم. (Perkins, 1998a)

نقطه ضعف واقعی IP، در روش آدرس‌دهی آن نهفته است. یک آدرس IP متشکل از یک شمارهٔ شبکه و یک شمارهٔ ماشین است. به عنوان مثال ماشینی با آدرس 160.80.40.20/16 را در نظر بگیرید. شمارهٔ شبکه 160.80 (یا 8272 در مبنای ده) و شمارهٔ ماشین 40.20 (یا 10260 در مبنای ده) است. مسیریابهای کل جهان یک جدول مسیریابی دارند که در آن خط مناسب برای رسیدن به شبکهٔ 160.80 مشخص شده است. وقتی بسته‌ای با آدرس IP بشکل 160.80.xxx.yyy، به مسیریاب وارد گردد بر روی آن خط ارسال خواهد شد.

بدین نحو اگر ماشینی با آدرس فوق به محل جدیدی نقل مکان کند، کماکان بسته‌های ارسالی برای او به شبکه یا مسیریاب خانگیش هدایت شده و از آن به بعد صاحب ماشین قادر به دریافت نامه‌های الکترونیکی یا نظائر آن نخواهد بود. انتساب آدرس IP جدید (متناظر با محل جدید) به ماشین سیّار، راهکار جالبی نیست چراکه این تغییر باید به تعداد زیادی از افراد، بانکهای اطلاعاتی و برنامه‌های کاربردی اطلاع داده شود.

راهکار دیگر آنست که مسیریابها در جدول مسیریابی به جای آنکه فقط شمارهٔ شبکه را نگه دارند آدرس کامل ماشینها را درج نمایند ولیکن این روش نیز مستلزم ذخیرهٔ میلیونها درایه (Entry) در جدول مسیریابی است و هزینه‌های نجومی به اینترنت تحمیل می‌کند.

از زمانی که تقاضا برای این قابلیت (که افراد بتوانند در هر کجا که هستند به اینترنت متصل شوند)، بالا گرفت، IETF یک گروه ویژه برای پیدا کردن راه حل مناسب تشکیل داد. این گروه کاری به سرعت اهدافی را که هر پیشنهاد یا راه حل باید آنها را برآورده می‌ساخت، تدوین و تعریف کردند. مهمترین این اهداف عبارت بود از:

۱. هر ماشین متحرک باید بتواند هر جا که هست از آدرس IP همیشگی و خانگی خود استفاده کند.

۲. هر گونه تغییر نرم‌افزاری در ماشینها مجاز شمرده نمی‌شود.

۳. هر گونه تغییر در نرم‌افزار مسیریابها یا جداول آنها مجاز نیست.

۴. اکثر بسته‌هایی که به سوی ماشینهای متحرک هدایت می‌شوند نباید از مسیر واقعی منحرف شوند.

۵. وقتی ماشین میزبان در محل همیشگی خود است نباید هیچ سربار اضافه‌ای تحمل کند.[1]

راه حل انتخابی همانی بود که در بخش ۵-۲-۸ آن را تشریح کردیم. در یک مرور اجمالی: هر سایتی که تمایل داشته باشد امکان سیار بودن را برای کاربران خود فراهم کند موظف به ایجاد یک «عامل خانگی» (Home Agent) است. همچنین هر سایتی که تمایل به پذیرش کاربران خارجی (کاربران سیار) داشته باشد موظف به ایجاد یک «عامل خارجی» (Foreign Agent) است. هر گاه یک ماشین متحرک در محدودهٔ یک سایت خارجی ظاهر می‌شود، ابتدا با عامل خارجی تماس برقرار کرده و ثبت نام می‌کند. عامل خارجی با عامل خانگی آن کاربر تماس گرفته و آدرسی جدید از محل کاربر متحرک به آن اعلام می‌کند. این آدرس عموماً آدرس IP خود عامل خارجی است.

وقتی بسته‌ای به LAN محل استقرار دائمی کاربر می‌رسد قبل از انتشار بر روی LAN به مسیریاب متصل به آن LAN وارد می‌شود. آن مسیریاب به روش معمول و همیشگی سعی می‌کند ماشین آن کاربر را پیدا کند لذا یک بستهٔ ARP منتشر کرده و مثلاً می‌پرسد: «آدرس اترنت ماشین 160.80.40.20 چند است؟». «عامل خانگی» به نیابت از کاربر، به این سؤال پاسخ داده و آدرس اترنت خود را اعلام می‌کند. از آن به بعد، مسیریاب تمام بسته‌های متعلق به 160.80.40.20 را برای «عامل خانگی» می‌فرستد. «عامل خانگی»، با مکانیزم ایجاد تونل بستهٔ اصلی را درون یک بستهٔ IP جدید جاسازی کرده و آدرس مقصد بستهٔ جدید را آدرس عامل خارجی قرار داده و آنرا ارسال می‌نماید. عامل خارجی پس از دریافت، بستهٔ اصلی را از درون آن استخراج کرده و آنرا به آدرس MAC ماشین متحرک (یعنی آدرس سخت‌افزاری تعریف شده در لایهٔ پیوند داده‌ها) ارسال می‌دارد. مضاف بر این، عامل خانگی کاربر، آدرس IP عامل خارجی را (که کاربر فعلاً مهمان اوست) به فرستندهٔ بسته‌ها اعلام می‌کند تا بسته‌های بعدی به کمک مکانیزم تونل مستقیماً به آدرس عامل خارجی ارسال شوند. این راه حل تمام نیازها و اهداف فوق‌الذکر را برآورده می‌کند.

اشاره به برخی از جزئیات ارزشمند است: در لحظه‌ای که ماشین متحرک، محل خود را ترک می‌کند احتمالاً مسیریاب در جدول نهان ARP[2] آدرس اترنت این ماشین را (که دیگر معتبر نیست) درج کرده و از آن استفاده می‌کند.[3] برای جایگزین کردن آدرس اترنت «ماشین عامل» به جای آدرس ماشین متحرک از راهکاری زیرکانه به نام «Gratuitous ARP» استفاده می‌شود. این روش مبتنی بر ارسال یک پیام خاص و ناخواسته برای مسیریاب است که باعث می‌شود یک درایهٔ دلخواه در جدول ARP به مقداری جدید تغییر کند. در اینجا درایه‌ای که باید تغییر کند متعلق به ماشین متحرک و در حال خروج از شبکه است. وقتی بعداً ماشین متحرک به شبکهٔ خود باز می‌گردد از همین راهکار برای بهنگام‌سازی مجدد جدول ARP در مسیریاب، استفاده می‌شود.[4]

هیچ مانعی وجود ندارد که یک ماشین متحرک نتواند «عامل خارجی» خودش باشد ولی این راهکار زمانی عملی است که ماشین متحرک در موقعیت جدید خود، به اینترنت دسترسی داشته باشد. در ضمن ماشین متحرک باید بتواند در محل جدید یک آدرس IP موقت جهت اعلام به «عامل خانگی» خودش بدست بیاورد. این آدرس

---

۱. یعنی اگر سربار کوچکی تحمیل می‌شود باید فقط زمانی باشد که ماشین میزبان در محل همیشگی خود نیست. -م

۲. ARP Cache

۳. یعنی چون پس از خروج ماشین متحرک، هنوز آدرس MAC آن در جدول ARP مسیریاب، وجود دارد تا موقعی که این آدرس اعتبار خود را حفظ می‌کند بسته‌ها به آدرسی که دیگر وجود ندارد ارسال خواهد شد. -م

۴. این راهکار آنست که «ماشین عامل» بدون آنکه هیچ سؤالی از او شده باشد یک بستهٔ پاسخ ARP (یعنی ARP Response) به صورت مصنوعی تولید و ارسال می‌کند. بدین ترتیب مسیریاب به اشتباه افتاده و محتوای آنرا در جدول خود درج می‌کند، هر چند هرگز در این خصوص سؤالی نپرسیده بود! -م

IP متعلق به LAN جدیدی است که ماشین متحرک موقتاً بدان متصل شده و مهمان آنست.

راه حل پیشنهادی IETF برای ماشینهای متحرک، مسائل متعدد دیگری را هم حل کرد که تا آن زمان بدان توجه نشده بود. به عنوان مثال یکی از مسائل این بود که ماشینهای عامل چگونه پیدا شوند؟ راه حل این مسئله آنست که هر عامل بطور متناوب آدرس خود و نوع سرویسهایی را که علاقمند به ارائه آنهاست، به صورت فراگیر منتشر نماید. وقتی ماشین متحرک به جایی وارد می‌شود ابتدا سعی می‌کند به این پیامهای انتشاری که اصطلاحاً «اعلان» (Advertisement) نامیده می‌شود، گوش فرا بدهد. گزینهٔ دیگر آنست که ماشین متحرک ورود خود را با انتشار بسته‌ای فراگیر (Broadcast) اعلام کرده و در انتظار پاسخ ماشین «عامل خارجی» باقی بماند.

مشکل دیگری که باید حل شود آنست که با ماشینهای متحرکی که با عجله و بدون خداحافظی شبکهٔ فعلی خود را ترک می‌کنند، چه باید کرد؟ راه حل آنست که حضور یک ماشین متحرک فقط برای مدت زمان ثابت و محدودی اعتبار داشته باشد. اگر ماشین متحرک بطور مرتب، اعتبار حضور خود را تمدید نکند و مهلت اعتبار او منقضی شود، عامل خارجی جداول خود را از مشخصات او پاک می‌نماید.

مشکل دیگر، موضوع امنیت است: وقتی یک عامل خانگی پیامی دریافت می‌کند که از او خواسته شده تمام بسته‌های متعلق به کاربری به نام روبرتا را به آدرس IP خاصی بفرستد، بهتر است این تقاضا پذیرفته نشود مگر آنکه اثبات شود که مبداء این درخواست، واقعاً روبرتا است نه کسی که خودش را به جای او جا زده است. بدین منظور از پروتکلهای احراز هویت مبتنی بر رمزنگاری استفاده می‌شود. این پروتکلها را در فصل هشتم تشریح خواهیم کرد.

آخرین نکته‌ای که گروه کاری IETF بدان پرداخت در خصوص «**سطوح تحرّک**» (Levels of Mobility) ماشینهاست. هواپیمایی را در نظر بگیرید که برای وصل کامپیوترهای مخصوص ناوبری و کنترل پرواز شبکه‌ای از نوع اترنت را بکار گرفته است. بر روی این شبکه یک مسیریاب استاندارد وجود دارد که از طریق یک لینک رادیویی با شبکهٔ اینترنت مستقر بر روی زمین در ارتباط است. از طرفی این ایده در ذهن برخی از افراد زیرک جرقه زده که در کنار دستهٔ صندلی هر مسافر یک کانکتور اترنت تعبیه شود تا مسافرین بتوانند در خلال پرواز، کامپیوترهای کیفی خود را به آن متصل کرده و به اینترنت وصل شوند. در این حالت باید برای کامپیوترهای متحرک دو سطح قائل شد: کامپیوترهای خود هواپیما، که نسبت به اترنت موجود در آن ثابت (Stationary) محسوب می‌شوند و کامپیوتر مسافران که نسبت به آن متحرک هستند. البته مسیریاب هواپیما نسبت به مسیریاب مستقر بر روی زمین، متحرک است. می‌توان متحرک بودن کامپیوترها نسبت به مسیریابهایی که خودشان متحرک هستند را از طریق ایجاد تونلهای متوالی پیاده‌سازی و اداره کرد.

۸-۶-۵　IPv6

اگرچه ممکن است مکانیزمهای CIDR و NAT چند سالی دیگر به دوام نسخهٔ چهارم IP کمک کنند ولی تقریباً بر همه آشکار شده که نفسهای پروتکل IP در شکل کنونی آن، به شماره افتاده است. مضاف بر مشکلات فنی IP، برخی از موارد پشت صحنه و زمینه‌ای دیگر نیز مطرح است. در سالهای اولیه، از اینترنت عموماً در دانشگاهها، صنایع پیشرفته و دولت ایالات متحده (خصوصاً وزارت دفاع) استفاده می‌شد. با گرایش بسیار زیاد مردم به اینترنت که از اواسط دههٔ نود شروع شد، گروههای مختلفی از افراد به آن رو آوردند؛ افرادی که نیازها و انتظارات متفاوتی داشتند. یکی از موارد آنست که افراد با کامپیوترهای بی‌سیم قابل حمل برای در ارتباط بودن با محل استقرار دائمی خود (ایستگاههای خانگی) می‌خواهند از اینترنت بهره بگیرند. مورد دیگر آن که با همگرایی قریب‌الوقوع صنایع کامپیوتر و مخابرات و صنایع تولید بازی و ابزار تفریح، دیری نخواهد پایید که حتی دستگاههای تلفن و تلویزیون در دنیا، به عنوان گرهی از اینترنت، به آن خواهد پیوست و در آن زمان میلیاردها

ماشین، از صدا و تصویر بهره خواهند گرفت. با در نظر داشتن چنین چشم‌اندازی ، IP بوضوح نیازمند تغییرات اساسی است و باید انعطاف بیشتری داشته باشد.

IETF که چنین افقی را پیش روی خود می‌دید در اوان ۱۹۹۰ کار را بر روی نسخهٔ جدیدی از پروتکل IP شروع کرد که در آن فضای آدرس هرگز با کمبود مواجه نشود و مشکلات عدیده‌ای را حل کند؛ قابلیت انعطاف بیشتری داشته باشد و در ضمن کارآمدتر باشد. اهداف عمدهٔ IPv6 عبارت بودند از:

۱. پشتیبانی از میلیاردها ماشین میزبان حتی در صورتی که تخصیص فضای آدرس ناکارآمد و با اسراف انجام شود.
۲. کاهش اندازهٔ جداول مسیریابی
۳. ساده‌سازی پروتکل به منظور افزایش سرعت پردازش مسیریابها
۴. ارائه امنیت بهتر در مقایسه با نسخهٔ فعلی IP (شامل احراز هویت و سرّی ماندن داده‌ها)
۵. توجه بیشتر به نوع خدمات و QoS ، به ویژه برای داده‌های بی‌درنگ
۶. کمک به فرآیند ارسال چندپخشی از طریق توصیف حوزه‌ها (Scopes)
۷. فراهم آوردن امکان جابجایی ماشینهای میزبان بدون تغییر در آدرس
۸. امکان ایجاد تغییر و پیشرفت در آینده
۹. امکان همزیستی پروتکلهای جدید و قدیم در طی سالها

برای توسعهٔ پروتکلی که تمام نیازهای فوق‌الذکر را برآورده نماید، IETF با انتشار RFC 1550 و در طی یک فراخوان، خواستار پیشنهادات دیگران در این خصوص شد. ۲۱ پیشنهاد دریافت گردید که اغلب آنها جامع نبودند. تا دسامبر ۱۹۹۲ فقط هفت طرح پیشنهادی قابل توجه در دستور کار قرار داشت. این طرحهای پیشنهادی از اصلاحات جزیی در نسخهٔ فعلی IP تا پیشنهاد دور انداختن آن و جایگزینی با یک پروتکل کاملاً متفاوت را شامل می‌شد.

یک پیشنهاد آن بود که TCP بر روی CLNP اجرا شود؛ پروتکلی که با آدرسهای ۱۶۰ بیتی فضای آدرس‌دهی نامحدود و جاویدان را فراهم کرده بود و دو پروتکل عمده و مهم لایهٔ شبکه را متحد و یکنواخت می‌کرد. ولیکن بسیاری افراد احساس کردند که پذیرش آن مهر تأییدی است بر این ادعا که هر کاری که OSI انجام داده صحیح تلقی می‌شود، داعیه‌ای که لااقل در حوزهٔ اینترنت به دلایل خاص نادرست است. الگوی CLNP بسیار شبیه به IP بود و این دو، تفاوت چندانی با هم ندارند. در آخر نیز طرحی انتخاب شد که تفاوت بسیار زیادی با IP و از آن بیشتر با CLNP دارد. ضربهٔ دیگری که CLNP خورد از آنجا بود که پشتیبانی ضعیفی از «نوع خدمات» (Type of Service) می‌کرد، خصوصیتی که برای انتقال کارآمد داده‌های چند رسانه‌ای به شدت نیاز بود.

سه تا از بهترین طرحهای پیشنهادی در ژورنال IEEE Network منتشر شد.[۱] پس از مباحثات فراوان، بازبینی ، ارزیابی موقعیت و سنجش استقبال عمومی، نسخه‌ای ترکیبی از طرحهای پیشنهادی Deering و Francis که SIPP[۲] نامیده می‌شد به عنوان طرح برگزیده معرفی و با عنوان IPv6 معرفی گردید.

IPv6 بخوبی اهداف مورد نظر را برآورده می‌کند: ویژگیهای خوب IP را نگه داشته، ویژگیهای بد را کنار گذاشته یا کمرنگ کرده و ویژگیهای جدیدی به آن افزوده است. بطور کلی IPv6 با IPv4 سازگار نیست ولی با تمام پروتکلهای جانبی اینترنت مثل TCP ، UDP ، ICMP ، IGMP ، OSPF ، BGP و DNS سازگار است. (البته

---

۱. Deering, 1993; Francis, 1993; and Katz and Ford, 1993

۲. Simple Internet Protocol Plus

ممکن است به دلیل آنکه آدرسها طولانی‌تر شده‌اند نیاز به اندکی تغییر داشته باشند.) ویژگیهای اساسی IPv6 در زیر تشریح شده است. برای آگاهی بیشتر در خصوص آن به RFCهای 2460 تا 2466 مراجعه نمایید.

اولین و مهمترین ویژگی آنست که IPv6 آدرسهای بسیار طولانی‌تری نسبت به IPv4 دارد. این آدرسها ۱۶ بایت طول دارند و دقیقاً مشکلی را حل کرده که به همان دلیل طراحی شد: یعنی تقریباً فضای نامحدودی از آدرسهای IP را فراهم آورده است. در این خصوص بیشتر صحبت خواهیم کرد.

دومین پیشرفت عمدهٔ IPv6 ساده‌سازی سرآیند آنست. این سرآیند جمعاً هفت فیلد دارد (در مقابل سیزده فیلد در IPv4). این تغییر، امکان آنرا فراهم آورده که مسیریاب بسته‌ها را سریعتر پردازش نماید و ظرفیت مفید مسیریاب را افزایش و تأخیر را کاهش دهد. در ادامه مختصراً به سرآیند خواهیم پرداخت.

سومین بهبود عمدهٔ آن پشتیبانی از گزینه‌های اختیاری (Options) است. این تغییر برای سرآیند جدید حیاتی بود چراکه برخی از فیلدهایی که در نسخهٔ قبلی وجودشان الزامی است در نسخهٔ فعلی اختیاریند. مضاف بر این، روش درج گزینه‌ها در این فیلد متفاوت از قبل است و اجازه می‌دهد مسیریابها بتوانند به سادگی از گزینه‌هایی که برایشان مهم نیست رد شوند. [یعنی در نسخهٔ جدید، دسترسی تصادفی و مستقیم به گزینه‌ها ممکن شده است.] این ویژگی سرعت پردازش بسته‌ها را افزایش می‌دهد.

چهارمین موضوعی که IPv6 در آن پیشرفت عمده‌ای داشته است، «امنیت» است. IETF با انبوهی از گزارشهای مطبوعاتی در خصوص نابغه‌های دوازده ساله‌ای مواجه بود که با کامپیوترهای شخصی خود، به شبکه‌های بانکی یا پایگاههای نظامی در سراسر اینترنت، نفوذ کرده بودند و جوّ شدیدی براه افتاده بود که باید برای بهبود امنیت شبکه‌ها کاری کرد. در نسخهٔ جدید IP ، «احراز هویت» (Authentication) و «حفظ امنیت اطلاعات» (Privacy) جزو ویژگیهای کلیدی به شمار می‌رود. البته این ویژگیها بعداً به IPv4 نیز افزوده شد [با عنوان IPsec]، فلذا در حال حاضر این دو ، در زمینهٔ امنیت تفاوت چندانی با هم ندارند.

موضوع آخر آنکه در نسخهٔ جدید به «کیفیت خدمات» (QoS) دقت بیشتری شده است. قبل از آن نیز تلاشهای جَسته و گریخته‌ای در این رابطه انجام شده بود ولیکن با رشد کاربردهای چند رسانه‌ای در اینترنت، این موضوع جدی‌تر و حساس‌تر به نظر می‌رسید.

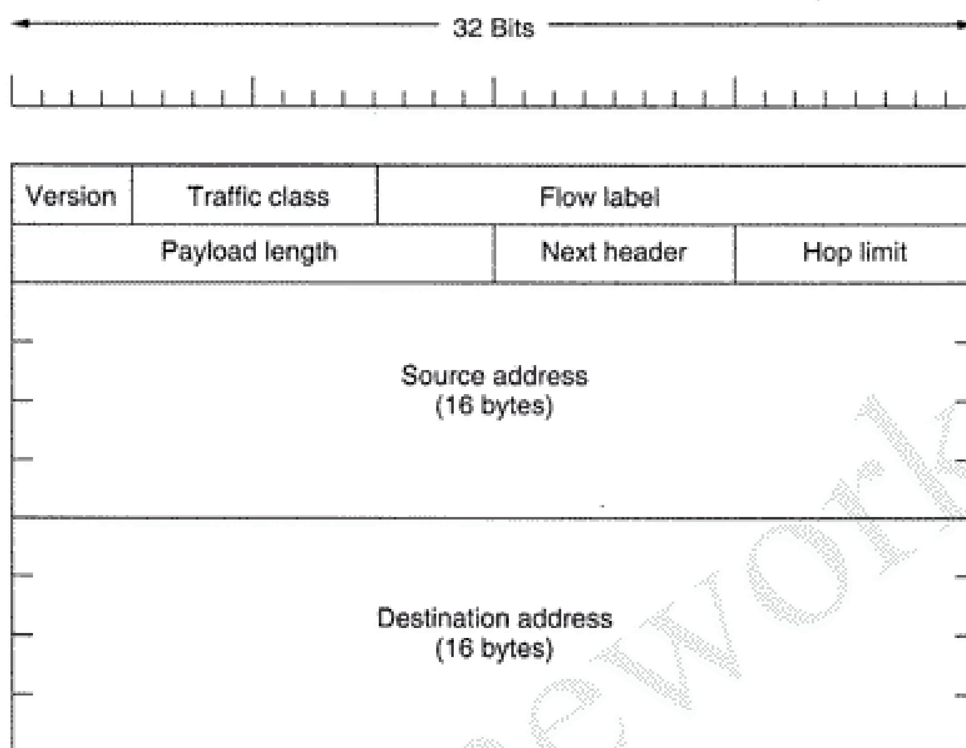
سرآیند اصلی IPv6

سرآیند اصلی IPv6 در شکل ۵-۶۸ نشان داده شده است. فیلد Version (شمارهٔ نسخهٔ پروتکل) برای IPv6 همیشه ۶ است. (کما اینکه برای IPv4 نیز همیشه ۴ است!). در دوران گذار از IPv4 به نسخهٔ جدید که ممکن است یک دهه طول بکشد، مسیریابها قادرند با بررسی این فیلد تشخیص بدهند که با چه نوع بسته‌ای روبرو هستند. البته از آنجایی که بررسی این فیلد به چندین دستورالعمل اجرایی CPU نیاز دارد و این کار زمان مفید پردازش هر بسته را هدر می‌دهد لذا در بسیاری از پیاده‌سازیهای عملی، برای اجتناب از این زمان تلفاتی، تشخیص آنکه یک بسته از نوع IPv4 است یا IPv6 ، با استفاده از فیلد خاصی در سرآیند لایهٔ پیوند داده‌ها بر عهدهٔ سخت‌افزار گذاشته شده است.[1] بدین ترتیب بسته‌ها براساس نوعشان مستقیماً به نرم‌افزار مناسب در لایهٔ شبکه هدایت می‌شوند. البته این الزام که لایهٔ پیوند داده از جزئیات نوع بسته‌های لایهٔ شبکه آگاه باشد با این اصل اساسی که «هر لایه نباید از معنای بیتهایی که از لایهٔ بالاتر تحویل او می‌شود، آگاه باشد» در تناقض است. بدون شک بحث و مناقشه بین طرفداران ایده‌های «انجام اصولگرایانه و صحیح کار» و «تسریع کار» به شدت ادامه خواهد داشت.

فیلد Traffic Class (کلاس ترافیک) برای تشخیص تفاوت بسته‌ها از لحاظ نیازمندیهای تحویل بی‌درنگ و

---

۱. مثلاً در فریم اترنت فیلد Type نوع بستهٔ درون فریم را تعیین می‌کند. ــم



شکل ۵-۶۸. سرآیند ثابت و الزامی در IPv6.

QoS درخواستی، بکار می‌آید. فیلدی با همین منظور از ابتدا در IP وجود داشت ولیکن استفاده از آن به صورت پراکنده و سلیقه‌ای بر روی مسیریابها پیاده‌سازی شد و اغلب مسیریابها آن را نادیده می‌گرفتند. اکنون تجربیات گذشته چراغ راهی شده تا بتوان بهترین راه و روش تحویل بسته‌های اطلاعات چندرسانه‌ای را تعیین کرد.

فیلد Flow Label (برچسب جریان) همچنان آزمایشی است ولی کاربرد مورد نظر آن، این بوده که بتوان یک **«شبه اتصال»** (Pseudoconnection) بین مبداء و مقصد، با ویژگیها و نیازمندیهای خاصّ ایجاد کرد. به عنوان مثال، جریانی از بسته‌ها که از یک پروسه در مبداء خاص تولید و به سوی یک پروسه بر روی مقصد خاص روانه می‌شوند احتمالاً نیاز به تضمین تأخیر محدود و مشخص دارد و در نتیجه باید پهنای باند لازم را رزرو کرد. در چنین مواردی می‌توان پیشاپیش یک «جریان» (Flow) با مشخصات درخواستی تنظیم کرد و به آن یک شناسه اختصاص داد. هر گاه مسیریاب بسته‌ای دریافت کند و فیلد Flow Label آن غیرصفر باشد، با مراجعه به جداول درونی خود تشخیص می‌دهد که با این بسته چگونه رفتار کند. در حقیقت استفاده از مفهوم «جریان»[۱] در IPv6 ، تلاشی است برای رسیدن به قابلیت انعطاف در زیرشبکه‌های دیتاگرام و تضمین کیفیت خدمات در زیرشبکه‌های مدار مجازی.

هویت هر «جریان» برحسب آدرس مبداء، آدرس مقصد و شمارهٔ جریان (برچسب جریان) مشخص می‌شود فلذا بین دو مبداء و مقصد در شبکه می‌توان بطور همزمان چندین «جریان» فعال تنظیم کرد. همچنین در این روش حتی اگر دو جریان متفاوت با شمارهٔ جریان یکسان از دو ماشین میزبان مختلف تولید و از مسیریابهای مشابهی عبور کنند، مسیریابها به کمک آدرس مبداء و مقصد قادر به تشخیص آنها خواهند بود. انتظار آنست که «برچسبهای جریان» به جای آنکه به صورت ترتیبی و از ۱ شروع شوند به صورت کاملاً تصادفی انتخاب گردند تا مسیریاب

۱. برای آشنایی با مفهوم جریان رجوع کنید به بخش ۵-۴-۱

بتواند آنها را در Hash Table خود درج کند.[۱]

فیلد Payload Length (طول قسمت حمل داده) مشخص می‌کند که پس از سرآیند ۴۰ بایتی در شکل ۵-۶۸ چند بایت داده قرار گرفته است. همین فیلد در IPv4 با نام Total Length وجود داشت. تغییر نام به آن دلیل بوده که در نسخهٔ جدید، سرآیند جزو طول بسته به حساب نمی‌آید بلکه فقط اندازه بخش حمل داده تعیین می‌شود.

فیلد Next Header ساختار بسته را سبکبار کرده است! دلیل آنکه سرآیند بسته ساده شده آنست که می‌توان در صورت لزوم سرآیند اضافی و انتخابی داشت. این فیلد مشخص می‌کند که پس از سرآیند ۴۰ بایتی کدامیک از سرآیندهای ششگانهٔ اضافی قرار گرفته است (در صورت وجود). اگر سرآیند اخیر، آخرین سرآیند بستهٔ IP باشد، این فیلد مشخص می‌کند که کدام پروسه در لایهٔ انتقال محتوای بسته را تحویل خواهد گرفت (مثلاً TCP، UDP و نظائر آن).[۲]

کاربرد فیلد Hop Limit آنست که بسته‌ها عمر محدودی داشته باشند. این فیلد در عمل مشابه با فیلد Time to Live (زمان حیات بسته) در IPv4 است یعنی به ازای عبور بسته از یک مسیریاب، یک واحد از مقدار آن کاسته می‌شود. در تئوری، مبنایی که IPv4 برای این فیلد در نظر داشت، «زمان بر مبنای ثانیه» بود در حالی که هیچ مسیریابی از چنین مبنایی استفاده نمی‌کند (بلکه به ازای هر گام یک واحد از آن می‌کاهد) لذا نام این فیلد را به گونه‌ای عوض کردند که عملکرد واقعی آن را نشان بدهد. هر گاه مقدار این فیلد در یک بسته به صفر برسد آن بسته حذف خواهد شد.

در ادامه فیلدهای Source Address و Destination Address (آدرس مبداء و مقصد) قرار گرفته‌اند. در طرح پیشنهادی آقای Deering آدرسها ۸بایتی انتخاب شده بودند در حالی که در مراحل بازبینی دیگران احساس کردند که شاید این فضای آدرس نیز در خلال چند دهه، IPv6 را نیز با کمبود فضای آدرس مواجه کند، در حالی که با آدرسهای ۱۶ بایتی هرگز چنین کمبودی رخ نخواهد داد. برخی از افراد معتقد بودند که آدرسهای ۱۶ بایتی بیش از حد بزرگ هستند در حالی برخی دیگر اعتقاد داشتند باید از آدرسهای ۲۰ بایتی استفاده شود تا با پروتکل دیتاگرام پیشنهادی OSI سازگار باشد. گروه دیگری نیز به آدرسهای با طول متغیر گرایش داشتند. پس از بحث و جدل فراوان، به این نتیجه رسیدند که آدرسهای با طول ثابت ۱۶ بایتی بهترین انتخاب است.

با توجه به طول زیاد آدرسهای IP ، نماد جدیدی برای نوشتن آنها پیشنهاد شد. این آدرسها به صورت هشت گروه که با علامت : از هم جدا شده، نوشته می‌شوند. هر گروه نیز به صورت چهار رقم هگزادیسمال نمایش داده‌می‌شود:

8000:0000:0000:0000:0123:4567:89AB:CDEF

از آنجایی که در آدرسها، تعداد ارقام صفر زیاد است، سه نوع بهینه‌سازی مجاز شمرده شده: صفرهای سمت چپ در هر گروه نوشته نمی‌شوند یعنی 0123 به صورت 123 نشان داده می‌شود؛ دوم آنکه اگر یک یا چند گروه شانزده بیتی تماماً صفر باشند با یک زوج علامت :: نشان داده می‌شود. بنابراین آدرس مثال بالا به صورت زیر نوشته خواهد شد:

8000::123:4567:89AB:CDEF

نهایتاً آنکه آدرسهای IPv4 را می‌توان با یک جفت :: و سپس آدرس نقطه‌دار قدیمی، نشان داد:

::192.31.20.46

---

۱. به عبارت دیگر مسیریاب انتظار دارد این شماره‌ها را Hash کند لذا این شماره‌ها نباید متوالی باشند.

۲. به عبارت دیگر محتوای این فیلد به صورت بازگشتی سرآیندهای بعدی را مشخص می‌کند تا نهایتاً به سرآیند آخر برسد که نوع بستهٔ لایهٔ انتقال را تعیین می‌نماید. –م

شاید لازم به گفتن نباشد که آدرسهای شانزده بایتی، فضایی معادل 2^{128} آدرس هستند که چنین فضایی تقریباً معادل 3×10^{38} آدرس است. اگر کل کرهٔ زمین شامل خشکیها و دریاها پر از کامپیوتر شوند باز هم IPv6 می‌تواند برای هر مترمربع 7×10^{23} آدرس IP فراهم کند. دانشجویان رشته شیمی می‌دانند که این عدد حتی از عدد آووگادرو نیز بزرگ است. [عدد آووگادرو $6/02\times10^{23}$ است]. چون در نظر نبوده که حتی به مولکولهای سطح زمین آدرس بدهیم آدرسهای IP شانزده بایتی، بهیچوجه کم نخواهد آمد!!

در عمل از فضای آدرس IP، بخوبی استفاده نخواهد شد. (دقیقاً همانند فضای شماره‌های تلفن که مثلاً فضای شماره‌های تلفن منهتن با پیش شمارهٔ ۲۱۲ پر شده ولی فضای شماره‌های ویومینگ (Wyoming) با پیش شمارهٔ ۳۰۷ تقریباً خالی مانده است.) دو نفر به نامهای Durand و Huitema در سند RFC 3194 محاسبه کرده‌اند که با ایده گرفتن از تخصیص شماره‌های تلفن و حتی در بدبینانه‌ترین حالت ممکن، باز هم می‌توان برای هر مترمربع از کرهٔ زمین، ۱۰۰۰ آدرس IP کنار گذاشت. در حالت کلی نیز می‌توان تریلیونها آدرس IP برای هر مترمربع از زمین در نظر گرفت. کوتاه سخن آنکه، در آینده هیچگاه به مشکل فضای آدرس برنخواهیم خورد.

مقایسهٔ سرآیند IPv4 (شکل ۵-۵۳) با سرآیند IPv6 (شکل ۵-۶۸) از این دیدگاه که چه فیلدی و چرا حذف شده است، آموزنده خواهد بود: فیلد IHL حذف شده زیرا سرآیند بسته‌های IPv6 طول ثابتی دارد. فیلد «پروتکل» وجود ندارد چرا که فیلد Next Header مشخص می‌کند که پس از آخرین سرآیند چه بستهٔ دیگری آمده است (بستهٔ TCP ، UDP یا نظائر آن).

تمام فیلدهایی که در ارتباط با «قطعه‌قطعه‌سازی» بسته‌ها در IPv4 تعریف شده بود در IPv6 حذف گردیده است زیرا پروتکل اخیر راهکار دیگری برای مکانیزم قطعه‌قطعه سازی برگزیده است. انتظار آنست که تمام ماشینهای سازگار با IPv6 بتوانند به صورت خودکار و پویا اندازهٔ دیتاگرامها را تعیین کنند و بدین نحو نیاز به قطعه‌قطعه شدن بسته‌ها کمتر اتفاق می‌افتد. همچنین حداقل طول بسته‌ای که هر ماشین موظف به پذیرش آنست از ۵۷۶ بایت به ۱۲۸۰ بایت افزایش یافته تا بتوان یک قطعه دادهٔ ۱۰۲۴ بایتی را به همراه تعداد زیادی سرآیند (۲۵۶ بایت)، بدون نیاز به قطعه‌قطعه شدن ارسال و دریافت کرد. مضاف بر این، وقتی ماشینی یک بستهٔ بیش از حد بزرگ IPv6 را ارسال می‌دارد مسیریاب ناتوان از هدایت آن، به جای قطعه‌قطعه کردن بسته آن را حذف کرده و پیام خطایی را باز می‌گرداند. این پیام به ماشین میزبان تفهیم می‌کند که باید بسته‌هایش را بشکند. البته اگر ماشین میزبان خودش بسته‌ها را با اندازهٔ مناسب ارسال کند کارآمدتر از آنست که بسته‌ها در طول مسیر شکسته شود.

فیلد Checksum نیز حذف شد زیرا محاسبهٔ آن کارآیی و سرعت پردازش بسته‌ها را بـه نحو چشمگیری کاهش خواهد داد. با توجه به قابلیت اعتماد شبکه‌های کنونی و با در نظر داشتن این حقیقت که در لایهٔ پیوند داده و لایهٔ انتقال نیز (بطور مجزا) صحت داده‌ها بررسی می‌شود، محاسبهٔ یک کد کشف خطای دیگر مثل checksum در مقایسه با کاهش کارآیی ارزشی ندارد. حذف این ویژگیهای زائد، IPv6 را به پروتکل متعادل و جمع و جور تبدیل کرده است. بدین ترتیب IPv6 به اهداف مورد نظر خود که همانا انعطاف، سرعت و فضای بزرگ آدرس بوده، نائل شده است.

سرآیندهای اضافی (سرآیندهای توسیع یا Extension Header)

گاهی به برخی از فیلدهای حذف شدهٔ IPv4 نیاز می‌شود و به همین منظور در IPv6 مفهوم جدیدی به نام «سرآیندهای توسیع» معرفی شده است. این سرآیندهای اختیاری برای افزودن اطلاعات به هر بسته بکار می‌آیند ولیکن روش کدینگ (و جاسازی) آنها کارآمد و سریع است. شش نوع مختلف سرآیند توسیع که تاکنون معرفی شده، در شکل ۵-۶۹ فهرست گردیده است. هر کدام از این سرآیندها اختیاریند ولیکن اگر به بیش از یک سرآیند نیاز باشد باید بطور پیاپی، پس از سرآیند ثابت و ترجیحاً به ترتیب فهرست، قرار بگیرند.

نام سرآیند توسیع (سرآیند اضافی)	توصیف عملکرد
Hop-by-hop options	حاوی اطلاعات گوناگون برای مسیریابها
Destination options	اطلاعات اضافی برای مقصد
Routing	فهرست ناکاملی از مسیریابها که بسته باید از آنها بگذرد.
Fragmentation	مدیریت قطعات دیتاگرام
Authentication	بررسی هویت فرستنده
Encrypted security payload	اطلاعاتی در خصوص محتوای رمزنگاری شده بسته

شکل ۵-۶۹. سرآیندهای توسیع در IPv6 (Extension Header).

برخی از سرآیندها دارای قالب ثابتی هستند در حالی که برخی دیگر تعداد متغیری فیلد با طول متفاوت دارند. به همین دلیل هر آیتم در قالب سه تایی **(نوع، طول، مقدار)**[۱] سازماندهی و کُد می‌شود. فیلد Type مشخص می‌کند که نوع گزینه چیست. مقدار فیلد **نوع** (Type) به نحوی انتخاب شده است که دو بیت ابتدایی آن به مسیریابهایی که نمی‌دانند آن گزینه را چگونه پردازش کنند، راه و چگونگی کار را نشان می‌دهند. این راهکارها عبارتند از: (۱) گزینهٔ مربوط را نادیده بگیر (۲) بسته را حذف کن (۳) بسته را حذف و یک بستهٔ ICMP برگردان (۴) بسته را حذف کن و یک بستهٔ ICMP برگردان ولیکن بستهٔ ICMP را برای آدرسهای «چندپخشی» (Multicast) نفرست. (تا یک بستهٔ چندپخشی اشتباه، منجر به تولید میلیونها گزارش ICMP نشود.)

فیلد یک بایتی **طول** (Length) مشخص می‌کند که فیلد **مقدار** (Value) چند بایتی است. (صفر تا ۲۵۵ بایت).

فیلد **مقدار** (Value) در برگیرنده اطلاعات مورد نیاز است و حداکثر می‌تواند ۲۵۵ بایت باشد.

«سرآیند توسیع Hop-by-Hop» (گام به گام) برای حمل اطلاعاتی کاربرد دارد که مسیریابهای واقع بر مسیر باید آنها را بررسی نمایند. قبلاً یکی از گزینه‌ها را معرفی کردیم: پشتیبانی از دیتاگرامهایی با طول بیش از ۶۴ کیلوبایت. قالب این سرآیند در شکل ۵-۷۰ نشان داده شده است. وقتی از این سرآیند استفاده می‌شود باید فیلد Payload Length (در سرآیند اصلی) به صفر مقداردهی شود.

Next header	0	194	4
Jumbo payload length			

شکل ۵-۷۰. «سرآیند توسیع Hop-by-Hop» (گام به گام) برای دیتاگرامهای بسیار بزرگ (جامبوگرام).

همانند تمام سرآیندهای اختیاری دیگر، این سرآیند نیز با فیلدی یک بایتی به نام Next Header شروع می‌شود و مشخص می‌کند که سرآیند بعدی از چه نوع است. پس از این بایت، بایت دیگری قرار گرفته که طول سرآیند Hop-by-Hop را بر مبنای بایت تعیین می‌کند ولیکن در مقدار آن، ۸ بایت ابتدایی (که وجود آن الزامی است) لحاظ نمی‌شود. تمام سرآیندهای توسیع دیگر نیز به همین نحو شروع می‌شوند. در ادامه فیلدی دو بایتی آمده که بایت اول مشخص می‌کند که این گزینه (Option) قرار است اندازهٔ دیتاگرام را تعریف کند (کد ۱۹۴) و بایت بعدی مشخص می‌کند که اندازهٔ دیتاگرام یک شمارهٔ چهار بایتی است. چهار بایت آخر این سرآیند، طول دیتاگرام را مشخص می‌کند. اندازهٔ زیر ۶۵۵۳۶ مجاز نیست و منجر به حذف بسته در اولین مسیریاب و بازگشت پیام خطای ICMP خواهد شد. دیتاگرامهایی که از این سرآیند اختیاری (یعنی Hop-by-Hop Header) استفاده

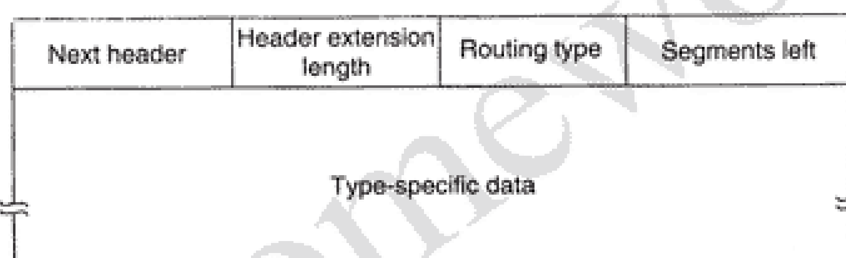
---

۱. (Type,Length,Value)

کرده‌اند اصطلاحاً Jumbogram (دیتاگرام عظیم) نامیده می‌شوند. [بدین ترتیب در IPv6 می‌توان قطعات دادهٔ بسیار بزرگ را به کمک سرآیند فوق به صورت یکجا ارسال کرد.] کاربرد جامبوگرامها در سوپرکامپیوترها که باید چندین گیگابایت اطلاعات را از طریق اینترنت منتقل کنند، بسیار حیاتی است.

«سرآیند توسیع Destination Options» برای درج فیلدهایی در نظر گرفته شده که صرفاً توسط ماشین مقصد پردازش و تفسیر می‌شوند. در نسخهٔ اولیهٔ IPv6، مقدار این گزینه پوچ در نظر گرفته شده و کاربردی نداشته است. وجود چنین فیلدی برای آن بوده که نرم‌افزار ماشینهای میزبان و مسیریابها چنین سرآیندی را به رسمیت بشناسند تا اگر روزگاری به آن نیاز شد، شرایط مهیا باشد وگرنه باید پروتکل عوض شود.

«سرآیند توسیع Routing»، فهرست مسیریابهایی را مشخص می‌نماید که بسته باید در راه رسیدن به مقصد از آنها عبور کند. این گزینه شباهت زیادی به گزینهٔ Loose Source Routing در IPv4 دارد. فهرست آدرسهایی که در این سرآیند مشخص شده باید در طول مسیر و به ترتیب ملاقات شوند ولی این امکان وجود دارد که مسیریابهایی هم که آدرس آنها در فهرست نیست مابین مسیر باشند. قالب سرآیند Routing در شکل ۵-۷۱ مشخص شده است.



شکل ۵-۷۱. سرآیند توسیع برای مسیریابی (Routing).

چهار بایت اول از این سرآیند، شامل چهار فیلد یک بایتی است: دو فیلد Next Header و Header Extension Length را قبلاً تعریف کردیم. فیلد Routing Type ، ساختار مابقی سرآیند را مشخص می‌نماید: مقدار صفر مشخص کنندهٔ آنست که پس از کلمهٔ چهار بایتی اول، یک کلمهٔ چهار بایتی دیگر قرار گرفته و پس از آن آدرس IPv6 (یعنی آدرسهای ۱۲۸ بیتی) مسیریابها قرار می‌گیرد. به غیر از این ساختار، فعلاً ساختار دیگری تعریف نشده مگر آنکه در آینده چیز جدیدی ابداع شود. فیلد آخر یعنی Segment Left تعداد آدرسهایی را مشخص می‌کند که هنوز ملاقات نشده‌اند. مقدار اولیهٔ این فیلد معادل با تعداد مسیریابهایی است که آدرس آنها در فهرست مورد نظر درج شده است و به ازای ملاقات در هر مسیریاب که آدرس آن در فهرست آمده یک واحد از این فیلد کاسته می‌شود. وقتی مقدار این فیلد در بسته به صفر برسد بسته روال طبیعی طی مسیر خود را از سر می‌گیرد بدون آنکه اجبار به عبور از مسیر خاصی داشته باشد. معمولاً در چنین لحظه‌ای بسته به مقصد خود نزدیک شده است.

سرآیند Fragmentation مشابه با IPv4 ، با مسئله قطعه‌قطعه سازی بسته‌ها سر و کار دارد. در این سرآیند نیز فیلدهای «شمارهٔ شناسایی دیتاگرام»، «شمارهٔ قطعه» و یک بیت MF تعریف شده‌اند که بیت MF مشخص می‌کند که آیا قطعهٔ جاری آخرین قطعهٔ دیتاگرام است یا آنکه قطعات دیگری در ادامه وجود دارند. البته در IPv6 برخلاف IPv4 ، فقط ماشین مبداء می‌تواند بسته‌ای را قطعه‌قطعه کند و مسیریابهای واقع بر روی مسیر قادر به چنین کاری نیستند. اگرچه این موضوع از لحاظ فلسفی یک واپسگرایی محسوب می‌شود ولی در عوض کار مسیریابها را ساده‌تر کرده و فرآیند مسیریابی سریعتر خواهد شد. همانگونه که قبلاً اشاره کردیم هر گاه یک مسیریاب با بسته‌ای

بیش از حد بزرگ مواجه گردد آن را حذف کرده و بستهٔ ICMP (حامل پیغام خطا و اطلاعات مفید دیگر) به مبداء آن بر می‌گرداند. اطلاعات ارسالی به مبداء بسته، امکان آنرا می‌دهد که به کمک این سرآیند، بسته را بـه قـطعات کوچکتر تقسیم و آنها را از نو ارسال کند.

سرآیند Authentication (سرآیند احراز هویت) مکانیزمی را فراهم آورده تا گیرنده بتواند از هویت فرستندهٔ بسته مطمئن شود. سرآیند Encrypted Security Payload اجازه می‌دهد تا محتوای بسته رمزنگاری شود و بدین ترتیب فقط گیرندهٔ مورد نظر قادر به خواندن آنست. این گونه سرآیندها برای انجام مأموریت خود از تکنیکهای رمزنگاری بهره می‌گیرند.

اختلاف نظرها و مناقشات

نظر به آنکه فرآیند طراحی IPv6، «باز» بوده و افراد درگیر در طراحی، بر عقاید خود تأکید داشته‌اند فلذا شگفت‌آور نیست که بسیاری از گزینه‌های انتخابی در IPv6 متناقض باشند. در زیر اجمالاً برخی از آنها را بررسی خواهیم کرد. برای آگاهی از جزئیات ماجرا به RFCهای مربوطه مراجعه نمائید.

قبلاً اشاره کردیم که بحث و جدل گسترده‌ای پیرامون طول آدرسها وجود داشت و توافق نهایی آن بـود کـه آدرسها با طول ثابت و ۱۶ بایتی باشند.

جدل دیگری بر سر حداکثر تعداد گام (Hop Limit) در گرفت. یک گروه احساس می‌کرد که محدود کردن حداکثر تعداد گام (Hop) به ۲۵۵ یک اشتباه محض است چرا که اگرچه در آن زمان حداکثر طول مسیرها عموماً از ۳۲ تجاوز نمی‌کرد ولی مدعی بودند که ممکن است ده سال بعد مسیرها طولانی‌تر از ۲۵۵ باشند. استدلال آنها این بود که فضای آدرس ۱۶ بایتی آینده‌نگری بیش از اندازه و در عوض مقدار کم Hop Count، کوته‌نظری است. از دیدگاه آنها بزرگترین اشتباه یک دانشمند کامپیوتر، آنست که برای هر فیلدی، تعداد بیت کمی در نظر بگیرد.

پاسخ گروه مقابل آن بود که افزایش بی‌مورد فضای هر فیلد منجر به تشکیل یک سرآیند حجیم خواهد شد. همچنین استدلال دیگرشان آن بود که وظیفهٔ فیلد Hop Count جلوگیری از سرگردانی بسته‌ها به مدت طولانی است و ۶۵۵۳۵ گام بیش از حد زیاد است. استدلال آخر آنکه با رشد اینترنت، لینکهای بسیار طولانی ساخته می‌شوند و این امکان فراهم می‌شود که برای رسیدن از یک کشور به کشور دیگر به کمتر از ده گام نیاز باشد. اگر یک بسته برای رسیدن از مبداء به مقصد مجبور شود از ۱۲۵ مسیریاب بین‌المللی بگذرد، ستون فقرات این شبکهٔ بین‌المللی در جایی اشکال دارد! بدین ترتیب طرفداران فیلد ۸ بیتی در عقیدهٔ خود پیروز شدند.

یکی دیگر از بحثهای داغ بر سر حداکثر طول بسته‌ها بود. سوپرکامپیوترها به بسته‌هایی با طول بیش از ۶۴ کیلوبایت احتیاج داشتند. وقتی یک سوپرکامپیوتر شروع به ارسال می‌کند و به کار خود مشغول می‌شود نباید بـه ازای هر ۶۴ کیلوبایت یکبار متوقف شود. استدلال گروه مخالف آن بود که اگر یک بستهٔ یک مگابایتی در طول مسیر به یک خط T1 برسد آن خط به مدت حداقل ۵ ثانیه اشغال شده و کاربران دیگری که در این خط سهیم هستند با تأخیر قابل توجهی روبرو خواهند شد.[1] توافق نهایی بدینجا ختم شد که بسته‌های معمولی حداکثر ۶۴ کیلوبایتی باشند ولی به کمک سرآیند اختیاری Hop-by-Hop بتوان جامبوگرامهایی با هر طول دلخواه ارسال کرد.

موضوع سوم مناقشه، حذف فیلد IPv4 checksum (کد تشخیص خطاهای احتمالی در سرآیند) بود. برخی از افراد حذف این فیلد را مشابه با برداشتن ترمزهای یک خودرو می‌دانستند که اگرچه ماشین را سبکبار و سریعتر می‌کند ولی اگر اتفاق غیرمترقبه‌ای رخ بدهد مشکل [illegible] بوجود می‌آید. [illegible] گروه مقابل آن بود که هر برنامهٔ کاربردی که نگران صحت داده‌های خود است باید از پروتکلی در لایهٔ انتقال بهره بگیرد که [illegible] داده‌ها را از لحاظ

۱. نرخ ارسال خط T1 را 1.544 مگابیت در نظر بگیرید.

سلامت بررسی می‌کند. لذا اضافه کردن کد کنترلی دیگر به لایهٔ IP برای کشف خطا (در حالی که هر بسته یکبار هم در لایهٔ پیوند داده بررسی می‌شود) بیهوده و زائد است. مضاف بر آن، تجربه نشان داده بود که محاسبهٔ جمع کنترلی (Checksum) در IPv4 هزینهٔ بالایی دارد. در این مناقشه نیز طرفداران حذف کد کشف خطا پیروز شدند.

موضوع دیگر، بحث ماشینهای همراه بود: وقتی یک کامپیوتر قابل حمل، در نیمی از کل دنیا حرکت می‌کند (مثلاً درون هواپیما)، آیا می‌تواند با همان آدرس IPv6 قبلی ، کار خود را ادامه بدهد یا آنکه مجبور به استفاده از ساختار «عامل خانگی» و «عامل خارجی» است؟ ماشینهای همراه مشکل «عدم تقارن» را به سیستم مسیریابی تحمیل می‌کنند: یک کامپیوتر همراه و کوچک براحتی قادر به شنیدن سیگنال قوی منتشره از مسیریاب ثابت خود هست ولی مسیریاب ثابت براحتی قادر به احساس سیگنال ضعیف ارسال شده توسط کامپیوتر همراه نیست. در نتیجه برخی از افراد گرایش داشتند که در IPv6 از ماشینهای همراه حمایت شود ولی تمام این تلاشها به دلیل آنکه بر روی هیچیک از طرحهای پیشنهادی توافقی بدست نیامد، با شکست مواجه گردید.

شاید بزرگترین مناقشه بر سر موضوع «امنیت» بود: همه بر این اصل که «امنیت لازم است» اشتراک نظر داشتند. دعوا بر سر چگونگی رسیدن به امنیت و محل پرداختن به آن بود. اولین محل پرداختن به امنیت لایهٔ شبکه است. استدلال موافقین مبنی بر آن بود که پیاده‌سازی امنیت در لایهٔ شبکه، سرویسی استاندارد فراهم می‌کند که تمام برنامه‌های کاربردی بدون هیچگونه برنامه‌ریزی قبلی می‌توانند از آنها بهره بگیرند. استدلال مخالفین نیز آن بود که برنامه‌های کاربردی امن، عموماً به هیچ مکانیزمی کمتر از رمزنگاری انتهابه‌انتها (End-to-End Encryption) احتیاج ندارند، به نحوی که پروسهٔ مبداء خودش داده‌های ارسالی خود را رمز کرده و پروسهٔ مقصد آنها را از رمز خارج کند. هر چیزی کمتر از این، می‌تواند کاربر را با خطراتی مواجه کند که از اشکالات امنیتی لایهٔ شبکه ناشی می‌شود و هیچ تقصیری از او نیست. پاسخ به این استدلال آن بود که کاربر می‌تواند امنیت لایهٔ IP را نادیده بگیرد و کار خودش را انجام بدهد! پاسخ نهایی مخالفین نیز آن بود که افرادی که به عملکرد صحیح شبکه (در خصوص امنیت) اعتماد ندارند چرا باید هزینهٔ پیاده‌سازی سنگین و کندی IP را بپردازند!!

یکی دیگر از جنبه‌های مربوط به امنیت این حقیقت بود که بسیاری از کشورها قوانین سختگیرانه‌ای در مورد صادرات محصولات مرتبط با رمزنگاری وضع کرده‌اند. از مثالهای بارز می‌توان به فرانسه و عراق اشاره کرد که حتی استفاده از رمزنگاری در داخل را نیز محدود کرده‌اند و عموم افراد نمی‌توانند چیزی را از پلیس مخفی نگه دارند. در نتیجه هر گونه پیاده‌سازی از IP که از روشهای رمزنگاری قوی استفاده می‌کند مجوز صدور از ایالات متحده (و بسیاری از کشورهای دیگر) را نخواهد گرفت. پیاده‌سازی دو نرم‌افزار یکی برای کاربرد داخلی و یکی برای صادرات، موضوعی است که عرضه‌کنندگان صنعت کامپیوتر با آن مخالفند.

موضوعی که پیرامون آن هیچ اختلاف نظر پیش نیامد آن بود که نمی‌توان انتظار داشت صبح روز یکشنبه IPv4 را در اینترنت از کار انداخت و صبح دوشنبه IPv6 را روشن نمود. در عوض مسیریابها و ماشینهایی که به IPv6 مجهز می‌شوند به مثابهٔ جزایر مستقل با استفاده از تونل با یکدیگر مبادلهٔ داده می‌کنند و با افزایش این جزایر، در یکدیگر ادغام شده و جزیرهٔ بزرگتری پدید می‌آید. در نهایت تمام این جزایر به هم می‌پیوندند و اینترنت کاملاً متحول می‌شود.

سرمایه‌گذاری حجیمی که از قبل بر روی مسیریابهای مبتنی بر IPv4 صورت گرفته، فرآیند تغییر و تحول اینترنت را سالها به تأخیر می‌اندازد. به همین دلیل تلاش زیادی صورت می‌گیرد تا این اطمینان حاصل شود که گذار از IPv4 به IPv6 حتی‌الامکان بدون زحمت و گرفتاری انجام گیرد. برای کسب آگاهی بیشتر در خصوص IPv6 به مرجع (Loshin, 1999) مراجعه نمایید.

۷ـ۵ خلاصه

لایهٔ شبکه به لایهٔ انتقال خدماتی را عرضه می‌کند. این لایه می‌تواند مبتنی بر «مدارات مجازی» یا «دیتاگرام» باشد. در هر دو حالت، وظیفهٔ اصلی این لایه آنست که بسته‌های داده را از مبداء به مقصد برساند. در «زیرشبکه‌های مدار مجازی»، تصمیم‌گیری در خصوص مسیر هدایت بسته‌ها، فقط یکبار و آن هم در حین تنظیم مدار مجازی انجام می‌شود. در «زیرشبکه‌های دیتاگرام»، این تصمیم‌گیری به ازای هر بسته تکرار می‌شود.

در شبکه‌های کامپیوتری از الگوریتمهای مسیریابی متنوعی استفاده می‌شود: الگوریتمهای ایستا نظیر «مسیریابی کوتاهترین مسیر» و «ارسال سیل‌آسا» (Flooding) و الگوریتمهای پویا شامل «مسیریابی بردار فاصله» و «مسیریابی حالت لینک» (Link State). در اغلب شبکه‌های واقعی بزرگ، از یکی از این دو روش مسیریابی پویا استفاده شده است. از موارد مهم دیگر در مسیریابی می‌توان به روش مسیریابی سلسله‌مراتبی، مسیریابی در محیط ماشینهای سیّار، مسیریابی پخش فراگیر، مسیریابی چندپخشی، و مسیریابی در شبکه‌های همتابه‌همتا اشاره کرد.

زیرشبکه‌ها ممکن است به سادگی با ازدحام مواجه شوند که منجر به افزایش تأخیر و کاهش ظرفیت مفید خروجی مسیریابها خواهد گردید. طراحان شبکه سعی می‌کنند با طراحی مناسب از بروز آن جلوگیری نمایند. این تکنیکها عبارتند از: تغییر در سیاستهای ارسال مجدد بسته‌ها، ذخیره‌سازی در حافظهٔ نهان، کنترل جریان و نظائر آنها. به هر حال اگر ازدحام رخ داد باید به نحوی به رفع آن اقدام کرد. می‌توان «بسته‌های دعوت به آرامش» (Choke Packet) پس فرستاد، بخشی از بار را دور ریخت و یا مکانیزمی نظیر به اینها را بکار بست.

مرحلهٔ بعدی پس از حل مسئله ازدحام، تلاش در تضمین کیفیت خدمات است. بدین منظور می‌توان از روشهایی نظیر «بافر کردن داده‌ها در ماشین مشتری»، «شکل‌دهی به ترافیک»، «رزرو منابع» و «کنترل پذیرش» استفاده کرد. راهکارهایی که برای تضمین کیفیت خوب خدمات طراحی و پیاده‌سازی شده عبارتند از: خدمات مجتمع (شامل RSVP)، خدمات متمایز و MPLS.

شبکه‌ها از جهات متعدد با هم تفاوت دارند، لذا وقتی می‌خواهند به یکدیگر متصل شوند مشکلاتی رخ خواهد داد. برخی از این مشکلات را می‌توان با تکنیک ایجاد تونل (Tunneling) از میان شبکهٔ واسط کاهش داد ولی اگر شبکه‌های مبداء و مقصد خودشان متفاوت باشند این راهکار نیز با شکست مواجه می‌شود. وقتی شبکه‌های متفاوت حداکثر طول بسته‌هایشان فرق کند به تکنیک قطعه‌قطعه‌سازی بسته‌ها نیاز خواهد شد.

در اینترنت مجموعهٔ وسیعی از پروتکلهای مرتبط با لایهٔ شبکه وجود دارد. از بین تمام اینها، پروتکل IP انتقال واقعی بسته‌ها را بر عهده دارد ولیکن پروتکلهای کنترلی دیگر مثل ICMP ، ARP و RARP و همچنین پروتکلهای مسیریابی مثل OSPF و BGP در کنار آن به فرآیند هدایت و انتقال بسته‌ها کمک می‌کنند. اینترنت سریعاً با مشکل کمبود فضای آدرس مواجه شد و برای رفع این مشکل نسخهٔ جدید IP یعنی IPv6 طراحی گردید.

مسائل

۱. دو مثال از برنامه‌های کاربردی کامپیوتر ارائه بدهید که برای آنها سرویس اتصال‌گرا (Connection Oriented) مناسب است. دو نمونهٔ دیگر از برنامه‌های کاربردی معرفی کنید که در آنها سرویس بدون اتصال مناسبتر است.

۲. آیا وضعیتی پیش می‌آید که در سرویس اتصال‌گرا بسته‌ها خارج از ترتیب تحویل مقصد شوند؟ پاسخ خود را تشریح کنید.

۳. زیرشبکه‌های دیتاگرام، هر بسته را به صورت واحدهای مجزا و مستقل از هم مسیریابی و هدایت می‌کنند. زیرشبکه‌های مدار مجازی بدین نحو عمل نمی‌کنند فلذا بسته‌های داده، مسیری از قبل تعیین‌شده را

می‌پیمایند. آیا این تعبیر بدان معناست که زیرشبکه‌های مدار مجازی به این قابلیت که بتوانند بسته‌های مجزا و مستقل را از یک مبداء دلخواه به مقصد مورد نظر هدایت کنند نیاز ندارند؟ پاسخ خود را شرح بدهید.

۴. سه نمونه از پارامترهایی را که می‌توان در حین تنظیم یک اتصال، بر روی آنها مذاکره و توافق کرد، نام ببرید.

۵. به مسئلهٔ زیر در خصوص پیاده‌سازی «سرویس مدار مجازی» دقت نمائید: اگر در درون زیرشبکه از الگوی مدار مجازی استفاده شده باشد هر بستهٔ داده باید یک سرآیند سه بایتی داشته باشد و هر مسیریاب نیز برای شناسایی هر مدار مجازی به ۸ بایت احتیاج دارد. اگر در درون زیرشبکه از الگوی دیتاگرام استفاده شده باشد، به سرآیندهای ۱۵ بایتی نیاز خواهد بود ولی در عوض مسیریاب به فضای حافظهٔ اضافی نیاز نخواهد داشت. هزینه ارسال را برای هر 10^6 بایت یک سنت (به ازای عبور از هر مسیریاب) فرض کنید. حافظه بسیار سریع لازم برای مسیریابها را یک سنت به ازای هر بایت در نظر بگیرید که با فرض ۴۰ ساعت کار در هر هفته، در عرض دو سال مستهلک می‌شود. بطور متوسط هر نشست هزار ثانیه طول می‌کشد و در خلال نشست ۲۰۰ بسته منتقل می‌شود.[1] هر بسته نیز بطور متوسط از چهار مسیریاب عبور می‌کند. پیاده‌سازی کدامیک از این الگوها ارزان‌تر تمام می‌شود و به چه میزان؟

۶. فرض کنید که تمام مسیریابها و ماشینها به درستی کار می‌کنند و تمام نرم‌افزارها از خطا مصون باشند. آیا باز هم احتمال آنکه یک بسته به اشتباه تحویل ماشینی دیگر شود وجود دارد؟ (حتی اگر این احتمال بسیار ناچیز باشد.)

۷. به شبکهٔ شکل ۵-۷ دقت کنید ولیکن وزنهای خطوط را نادیده بگیرید. فرض کنید مسیریابی طبق الگوریتم سیل‌آسا (Flooding) انجام می‌شود. اگر بسته‌های ارسالی از A به D دارای «شمارهٔ گام» (Hop count) معادل ۳ باشند فهرست مسیرهایی را که بسته از آنها عبور می‌کند، مشخص نمائید. همچنین تعیین کنید که کل گامهایی که بسته‌ها پیموده و پهنای باندی که به ازای آن تلف می‌شود چقدر است.

۸. روشی ساده و ذهنی برای پیدا کردن دو مسیر از یک مبداء مشخص به مقصدی خاص در شبکه ارائه بدهید به نحوی که بتواند در مقابل از بین رفتن خطوط انتقال، دوام بیاورد. (با فرض آنکه حداقل دو مسیر وجود دارد). مسیریابها را قابل اعتماد فرض کنید لذا نگران از کار افتادن آنها نباشید.

۹. به زیرشبکه شکل ۵-۱۳-الف دقت کنید. در این زیرشبکه از «مسیریابی بردار فاصله» استفاده شده و بردارهای ذیل توسط مسیریاب C دریافت شده است:

از B : (5,0,8,12,6,2)   از D : (16,12,6,0,9,10)   از E : (7,6,3,9,0,4)

تأخیر اندازه‌گیری شده تا B و D و E به ترتیب ۶ و ۳ و ۵ است. جدول جدید مسیریابی در C چیست؟ خط خروجی و تأخیر تخمینی رسیدن به هر مقصد را مشخص نمائید.

۱۰. اگر میزان تأخیر به صورت اعداد هشت بیتی ذخیره شود و شبکه ۵۰ مسیریاب داشته باشد و بردارهای تأخیر در هر ثانیه دو بار مبادله شوند چه مقدار از پهنای باند یک خط دو طرفهٔ همزمان (Full Duplex)، در اثر استفاده از الگوریتم مسیریابی توزیع شده [بردار فاصله] تلف می‌شود؟ فرض کنید هر مسیریاب سه خط با مسیریابهای دیگر دارد.

۱۱. در شکل ۵-۱۴، حاصل OR کردن دو مجموعهٔ بیتهای ACF با یکدیگر ۱۱۱ می‌شود. آیا این موضوع تصادفی است یا در هر زیرشبکه و با هر وضعیتی رخ می‌دهد؟

۱۲. برای مسیریابی سلسله‌مراتبی در زیرشبکه‌ای با ۴۸۰۰ مسیریاب، اندازهٔ هر منطقه (Region) و دسته (Cluster) چقدر باشد تا جدول مسیریابی (در سلسله‌مراتب سه‌سطحی) حداقل شود؟ نقطهٔ شروع مناسب

---

۱. نشست را در یک عبارت غیرفنی می‌توانید ارتباط و محاورهٔ یک زوج ماشین در طول زمان فرض کنید.

آنست که فرض کنید داشتن k دسته و k منطقه و k مسیریاب تقریباً بهینه است، فلذا k تقریباً ریشهٔ سوم ۴۸۰۰ (تقریباً ۱۶) می‌شود. با سعی و خطا در همسایگی عدد ۱۶، بهترین تعداد دسته، منطقه و مسیریاب را پیدا کنید.

۱۳. در متن گفته شد که وقتی ماشین متحرک در محل همیشگی خود نیست، بسته‌های ارسالی به سوی LAN خانگی، متوقف و به سمت LAN موقت او تغییر مسیر داده می‌شود. در یک شبکهٔ IP که بر روی اترنت پیاده شده، این تغییر مسیر چگونه انجام می‌شود؟

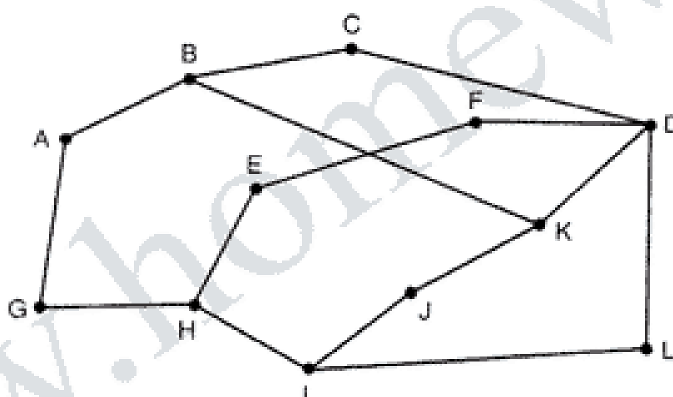
۱٤. با در نظر گرفتن زیرشبکه شکل ۵-۶، اگر B بسته‌ای را به صورت فراگیر پخش کند، چه تعداد بسته در زیرشبکه تولید می‌شود اگر:

الف) اگر از روش هدایت در مسیر معکوس (Reverse Path Forwarding) استفاده شود؟

ب) از روش Sink Tree استفاده شود؟

۱۵. شبکهٔ شکل ۵-۱۶-الف را مدنظر قرار بدهید. فرض کنید که خط جدیدی بین F و G اضافه می‌شود ولی کماکان Sink Tree شکل ۵-۱۶-الف بی‌تغییر باقی می‌ماند. در شکل ۵-۱۶-ج چه تغییری حاصل می‌شود؟

۱۶. برای زیرشبکه بعدی درخت پوشای چندپخشی (Multicast Spanning Tree) را برای مسیریاب C ترسیم کنید به نحوی که اعضای گروه مورد نظر مسیریابهای A و B و C و D و E و F و I و K باشند.



۱۷. در شکل ۵-۲۰، آیا گره‌های H یا I در جستجویی که از مبداء A شروع شده، هیچگونه ارسال فراگیر داشته‌اند؟

۱۸. در شکل ۵-۲۰ فرض کنید که گره B از نو راه‌اندازی شده است و هیچگونه اطلاعات مسیریابی در جدول خود ندارد. این گره به ناگاه احتیاج به یافتن مسیری به H پیدا می‌کند و بسته‌هایی پخشی (Broadcast) با مقدار TTL معادل ۱ و ۲ و ۳ و... را منتشر می‌کند. این کار چند دور طول می‌کشد تا نهایتاً مسیری پیدا شود؟

۱۹. در ساده‌ترین نسخهٔ الگوریتم Chord برای «جستجوی همتابه‌همتا» (Peer-to-Peer)، برای جستجو از جدول Finger استفاده نمی‌شود و به جای آن، جستجو به صورت خطی (یعنی گره‌به‌گره) در دو جهت بر روی دایره انجام می‌گیرد. آیا یک گره می‌تواند به درستی حدس بزند که جستجو را باید در کدام یک از جهات انجام بدهد؟ پاسخ خود را شرح بدهید.

۲۰. دایرهٔ Chord نشان داده شده در شکل ۵-۲۴ را مدنظر قرار بدهید. فرض کنید که گره ۱۰ به ناگاه فعال و وارد خط می‌شود. آیا ورود این گره در جدول Finger از گره ۱ تأثیری می‌گذارد و اگر این چنین است چگونه؟

۲۱. به عنوان یک مکانیزم کنترل ازدحام در زیرشبکه‌ای که در درون از مدار مجازی بهره گرفته است، هر مسیریاب می‌تواند از اعلام وصول یک بسته (Ack) طفره برود مگر آنکه: (۱) آگاه شود که آخرین ارسال او

بر روی مدار مجازی، با موفقیت دریافت شده است و (۲) بافر آزاد داشته باشد. برای سادگی فرض کنید مسیریاب برای اعلام وصول از پروتکل «توقف و انتظار» (Stop & Wait) استفاده کرده و برای هر مدار مجازی یک بافر اختصاصی (برای ترافیک هر یک از دو جهت) اختصاص داده است. اگر انتقال هر بسته T ثانیه طول بکشد (انتقال بسته یا پیام Ack به یک اندازه طول می‌کشد) و n مسیریاب بر روی مسیر قرار گرفته باشد، در چنین شرایطی بسته‌ها با چه نرخی تحویل ماشین مقصد می‌شوند؟ فرض کنید که خطای انتقال بسیار ناچیز و اتصال بین ماشین و مسیریاب بی‌نهایت سریع است.

۲۲. در یک زیرشبکه دیتاگرام به مسیریابها اجازه داده شده تا در صورت نیاز بسته‌ها را حذف نمایند. احتمال حذف بسته در هر مسیریاب را p فرض کنید. حالتی را در نظر بگیرید که ماشین مبداء به مسیریاب مبداء، مسیریاب مبداء مستقیماً به مسیریاب مقصد و مسیریاب مقصد نیز مستقیماً به ماشین مقصد متصل شده است. اگر یکی از این مسیریابها بسته‌ای را حذف نمایند، نهایتاً مهلت ماشین مبداء به سر می‌رسد و آن را از نو ارسال می‌نماید. اگر خط بین ماشین و مسیریاب را مثل خط بین دو مسیریاب، یک «گام» (Hop) فرض کنیم، مقدار هر یک از موارد زیر را حساب کنید:

الف) میانگین تعداد گامی که یک بسته در هر بار ارسال طی می‌کند.

ب) میانگین دفعات ارسال یک بسته

ج) میانگین گامهایی که یک بستهٔ دریافتی طی کرده است.

۲۳. دو تفاوت عمدهٔ روش «بیت هشدار» (Warning Bit) و روش RED را توضیح بدهید.

۲۴. دلیلی ارائه بدهید که چرا در الگوریتم «سطل سوراخ»، در هر تیک ساعت مجوز ارسال یک بسته صادر می‌شود، فارغ از آنکه طول بسته چقدر است.

۲۵. گونه‌ای از الگوریتم سطل سوراخ که مبتنی بر شمارش بایت است در سیستم خاصی بکار گرفته شده است. قاعده آنست که در هر تیک ساعت یک بستهٔ ۱۰۲۴ بایتی یا دو بستهٔ ۵۱۲ بایتی یا معادل آن، ارسال شود. یکی از محدودیتهای جدی چنین سیستمی را که در متن درس بدان نپرداخته‌ایم، بیان کنید.

۲۶. در یک شبکهٔ ATM از روش سطل نشانه‌دار برای شکل‌دهی به ترافیک استفاده شده است. در هر پنج میکروثانیه یک توکن جدید در سطل قرار داده می‌شود. هر توکن برای ارسال یک سلول حاوی ۴۸ بایت داده، مناسب می‌باشد. حداکثر نرخ ارسال مجاز چقدر است؟

۲۷. ترافیک یک کامپیوتر متصل به شبکهٔ 6 Mbps طبق الگوریتم سطل نشانه‌دار، شکل‌دهی و منظم می‌شود. سطل نشانه‌دار با نرخ یک مگابیت بر ثانیه پر می‌شود. ظرفیت سطل هشت مگابیت است که در همان ابتدا پر شده است. کامپیوتر در چه مدت می‌تواند با سرعت ۶ مگابیت بر ثانیه ارسال داشته باشد؟

۲۸. فرض کنید در توصیف مشخصات یک «جریان» حداکثر طول هر بسته ۱۰۰۰ بایت، نرخ سطل نشانه‌دار ده میلیون بایت در ثانیه، اندازهٔ سطل نشانه‌دار یک میلیون بایت و حداکثر نرخ ارسال ۵۰ میلیون بایت در ثانیه تعیین شده است. یک ماشین در چه مدتی قادر به ارسال انفجاری داده‌ها با حداکثر سرعت خود می‌باشد؟

۲۹. شبکهٔ شکل ۵-۳۷ از RSVP و درخت چندپخشی نشان داده شده برای ماشینهای ۱ و ۲، بهره گرفته است. فرض کنید که ماشین ۳ برای دریافت یک «جریان» از ماشین ۱، تقاضای کانالی با پهنای باند 2MB/sec می‌دهد. همچنین برای دریافت جریانی از ماشین ۲ تقاضای 1MB/sec می‌دهد. بطور همزمان ماشین ۴، 2MB/sec برای دریافت جریان از ماشین ۱ و همچنین ماشین ۵ برای دریافت جریان از ماشین ۲، 1MB/sec پهنای باند درخواست می‌کنند. مجموع کل پهنای باند رزرو شده در مسیریابهای A و B و C و E و H و J و K و L برای این درخواستها چقدر است؟

۳۰. پردازندهٔ یک مسیریاب قادر به پردازش ۲ میلیون بسته در هر ثانیه است. بار عرضه شده به آن ۱/۵ میلیون بسته در ثانیه است. اگر در مسیر رسیدن از مبداء به مقصد، ۱۰ مسیریاب قرار گرفته باشند، تأخیر ناشی از صف‌بندی و پردازش بسته چقدر است؟

۳۱. فرض کنید کاربری از خدمات متمایز و «هدایت پرشتاب» بهره می‌گیرد. آیا تضمینی وجود دارد که بسته‌های پرشتاب تأخیری کمتر از بسته‌های معمولی داشته باشند؟ دلیل خود را تشریح کنید.

۳۲. آیا به فرآیند قطعه‌قطعه‌سازی بسته‌ها در شبکه‌های بهم متصل شدهٔ مدار مجازی نیز احتیاج است یا آنکه این نیاز فقط در شبکه‌های دیتاگرام وجود دارد؟

۳۳. فرآیند ایجاد تونل از میان یک شبکهٔ الحاق شده مدار مجازی ساده و سرراست است: مسیریاب چند پروتکلی در یکی از دو طرف یک مدار مجازی با مسیریاب طرف دیگر ایجاد و تنظیم می‌کند و بسته‌ها را از طریق این مدار مجازی مبادله می‌نماید. آیا ایجاد تونل از میان یک شبکهٔ دیتاگرام نیز ممکن است؟ اگر جواب مثبت است چگونه؟

۳۴. فرض کنید که ماشین میزبان A به مسیریاب R1 ، R1 نیز به مسیریاب دیگری به نام R2 ، و R2 نیز به ماشین میزبان B متصل است. فرض کنید که یک پیام TCP حاوی ۹۰۰ بایت داده و ۲۰ بایت سرآیند، جهت تحویل به B به نرم‌افزار IP در ماشین A تسلیم می‌شود. محتوای فیلدهای Total Length ، Identification ، DF، MF و Fragment offset را در هر بستهٔ IP که از یکی از سه لینک A-R1 ، R1-R2 ، R2-B عبور می‌کند، مشخص نمایید؛ لینک A-R1 می‌تواند از فریمی با طول حداکثر ۱۰۲۴ بایت (که ۱۴ بایت آن هم سرآیند فریم لایهٔ پیوند داده‌ها محسوب می‌شود) پشتیبانی می‌کند. لینک R1-R2 می‌تواند از فریمی با طول ۵۱۲ بایت حمایت کند (با احتساب ۸ بایت سرآیند فریم) و لینک R2-B از فریمهایی با طول حداکثر ۵۱۲ بایت (با احتساب ۱۲ بایت سرآیند فریم) پشتیبانی می‌کند.

۳۵. یک مسیریاب بسته‌های IP با طول ۱۰۲۴ بایت (با احتساب داده و سرآیند) در خروجی خود گسیل می‌دارد. با فرض آنکه بسته‌ها فقط برای ده ثانیه زنده می‌مانند، مسیریاب حداکثر با چه سرعتی می‌تواند بسته‌ها را بفرستد بدون آنکه خطر به صفر برگشتن فیلد شمارهٔ شناسایی دیتاگرام (Datagram ID) وجود داشته باشد؟

۳۶. یک دیتاگرام IP که از گزینهٔ Strict Source Routing استفاده کرده، مجبور است قطعه قطعه شود. به نظر شما آیا این گزینه بایستی در تمام قطعات آن قرار داده شود یا آنکه قرار دادن این گزینه در قطعهٔ اول کفایت می‌کند؟ پاسخ خود را شرح بدهید.

۳۷. فرض کنید در کلاس B به جای ۱۶ بیت برای شمارهٔ شبکه ، ۲۰ بیت در نظر گرفته می‌شد. در چنین حالتی چند شبکهٔ کلاس B می‌توان داشت؟

۳۸. آدرس IP با نمایش هگزادیسمال C22F1582 را به نماد نقطه‌دار تبدیل کنید.

۳۹. شبکه‌ای در اینترنت، از الگوی زیرشبکه 255.255.240.0 استفاده کرده است. این شبکه حداکثر چند ماشین میزبان می‌تواند داشته باشد؟

۴۰. تعداد بسیار زیادی آدرس IP متوالی از نقطهٔ شروع 198.16.0.0 در اختیار می‌باشد. فرض کنید چهار سازمان A و B و C و D به ترتیب ۴۰۰۰ ، ۲۰۰۰ ، ۴۰۰۰ و ۸۰۰۰ آدرس IP درخواست می‌کنند. برای هر یک از اینها اولین آدرس IP ، آخرین آدرس IP و الگوی زیرشبکه را به شکل w.x.y.z/s ، معین کنید.

۴۱. یک مسیریاب آدرسهای IP جدیدی طبق فهرست زیر دریافت می‌کند (جهت درج در جدول مسیریابی):
57.6.96.0/21 و 57.6.104.0/21 و 57.6.112.0/21 و 57.6.120.0/21

اگر برای رسیدن به تمام این شبکه‌ها از خط مشترکی استفاده شود، آیا می‌توان این آدرسها را «تجمیع»

(Aggregate) کرد؟ اگر می‌توان به چه نحو؟ اگر خیر، چرا؟

۴۲. مجموعه‌ای از آدرسهای IP از 29.18.0.0 تا 29.18.128.255 به صورت 29.18.0.0/17 «تجمیع» شده‌اند. ولیکن یک فاصلهٔ خالی ۱۰۲۴ تایی منتسب نشده در محدودهٔ 29.18.60.0 تا 29.18.63.255 وجود داشته که به ناگاه به شبکه‌ای اختصاص داده می‌شود که خط خروجی رسیدن به آن، با بقیه فرق می‌کند. آیا در اینجا باید فضای تجمیع شدهٔ آدرسها را مثل اول به بلوکهای اولیه تقسیم کرد و بلوک جدید به جدول اضافه و فرآیند تجمیع از نو انجام گیرد؟ اگر نه چه کار دیگری می‌توان انجام داد؟

۴۳. یک مسیریاب در جدول مسیریابی خود درایه‌های CIDR زیر را در اختیار دارد:

Address/mask (آدرس/الگوی زیرشبکه)	Next Hop (گام بعدی)
135.46.56.0/22	Interface 0
135.46.60.0/22	Interface 1
192.53.40.0/23	Router 1
default	Router 2

اگر بسته‌ای با یکی از آدرسهای IP زیر دریافت شود، مسیریاب با آن بسته چه می‌کند:

الف) 135.46.63.10
ب) 135.46.57.14
ج) 135.46.52.2
د) 192.53.40.7
ه) 192.53.56.7

۴۴. سیاست بسیاری از شرکتها بر آنست که دو (یا چند) مسیریاب متصل به اینترنت در شبکهٔ خود داشته باشند تا در صورت از کار افتادن یکی از آنها، از دیگری استفاده شود. آیا با چنین سیاستی باز هم می‌توان از NAT بهره گرفت؟ پاسخ خود را تشریح کنید.

۴۵. فرض نمائید پروتکل ARP را برای دوستتان توضیح داده‌اید. پس از اتمام توضیحاتتان او می‌گوید: «متوجه شدم: ARP سرویسی را به لایهٔ شبکه ارائه می‌دهد و طبعاً جزئی از لایهٔ پیوند داده‌ها است!» چه حرفی برای گفتن به او دارید؟

۴۶. ARP و RARP هر دو آدرسهایی را از یک فضا به فضایی دیگر می‌نگارند (IP به MAC و بالعکس). از این دیدگاه هر دو مشابه یکدیگر هستند، ولی پیاده‌سازی آنها متفاوت از یکدیگر است. عمده‌ترین تفاوت آنها در چیست؟

۴۷. روشی را برای بازسازی قطعات یک بستهٔ IP در مقصد، معرفی و تشریح نمائید.

۴۸. اغلب الگوریتمهای بازسازی دیتاگرامهای IP ، دارای زمان‌سنج (تایمر) خاصی هستند تا در صورت از بین رفتن یک قطعه، بقیهٔ قطعات تا ابد در بافر نمانند. فرض کنید یک دیتاگرام به چهار قطعه تقسیم شده است. سه قطعهٔ اول سر موقع دریافت می‌شوند ولی قطعهٔ آخر با تأخیر مواجه می‌شود. عاقبت مهلت دریافت آن منقضی شده و سه قطعهٔ دیگر از حافظهٔ گیرنده، پاک می‌گردد. اندکی بعد، آخرین قطعه از راه می‌رسد. گیرنده با آن چه باید بکند؟

۴۹. چه در IP و چه در ATM ، کد کشف خطای checksum فقط سرآیند بسته را در بر می‌گیرد نه بخش داده را. فکر می‌کنید منطق این طراحی چه بوده است؟

۵۰. شخصی که در بوستون زندگی می‌کند در سفری به مینیاپولیس کامپیوتر کیفی خود را به همراه می‌برد. خوشبختانه، شبکهٔ LAN او در مینیاپولیس یک شبکهٔ محلی بی‌سیم و مبتنی بر IP است و او مجبور به وصل

کابل به جایی نیست. آیا برای آنکه ترافیک پست الکترونیکی یا دیگر داده‌ها به درستی تحویل او شود، کماکان لازم است که از دو عامل خانگی و خارجی استفاده شود؟

۵۱. آدرسها در IPv6 شانزده بایتی هستند. اگر در هر پیکوثانیه یک بلوک یک میلیون تایی آدرس IP رزرو شود چه مدت طول می‌کشد تا آدرسها تمام شوند؟

۵۲. فیلد پروتکل که در IPv4 وجود داشت اکنون در سرآیند ثابت IPv6 وجود ندارد. به چه دلیل؟

۵۳. آیا وقتی که پروتکل IPv6 به صحنهٔ عمل آمد نیازی به تغییر در ARP هست؟ اگر این چنین است آیا این تغییرات در عملکرد و مفهوم است یا در پیاده‌سازی فنی؟

۵۴. برنامه‌ای بنویسید که فرآیند مسیریابی به روش سیل‌آسا را شبیه‌سازی نماید. هر بسته دارای یک فیلد شمارنده است که به ازای هر گام (Hop) یک واحد از آن کم می‌شود. وقتی این شمارنده به صفر برسد، بسته حذف می‌گردد. زمان را «گسسته» (Discrete) فرض کنید و در هر واحد زمان فقط یک بسته بر روی خط ارسال یا دریافت می‌شود. برنامهٔ خود را در سه نسخه تهیه کنید: (۱) بستهٔ ورودی بر روی تمام خطوط ارسال گردد. (۲) بستهٔ ورودی بر روی تمام خطوط به استثنای خطی که از آن داخل شده، فرستاده شود. (۳) بستهٔ ورودی بر روی k تا از بهترین خطوط (به صورت آماری و غیردقیق) ارسال گردد. روش سیل‌آسا را با روش معمولی (یعنی وقتی بستهٔ ورودی بر روی بهترین خط خروجی ارسال می‌شود) از لحاظ پهنای باند و تأخیر مقایسه نمائید.

۵۵. برنامه‌ای بنویسید که یک شبکهٔ کامپیوتری را به صورت زمان گسسته شبیه‌سازی کند. در هر واحد زمان، بسته‌ای که سر صفّ هر یک از خطوط خروجی مسیریابها قرار گرفته‌اند، ارسال و یک گام جلو می‌روند. هر مسیریاب فضای بافر محدودی دارد. اگر بسته‌ای دریافت شود ولی جایی برای آن وجود نداشته باشد، حذف شده و از نو ارسال نخواهد شد. در عوض یک پروتکل انتها به انتها (End to End) وجود دارد که در صورت ارسال داده‌ای که دریافت آن تصدیق نشده از مبداء ، داده‌ها را مجدداً ارسال می‌کند. توان مفید (ظرفیت مفید یا Throughput) شبکه را برحسب تابعی از زمان انقضای مهلت (Timeout) و پارامتر نرخ خطا ترسیم نمائید.

۵۶. تابعی بنویسید که عملیات هدایت (Forwarding) را در یک مسیریاب مبتنی بر IP انجام بدهد. این تابع یک پارامتر ورودی دارد و آن هم آدرس IP است. فرض کنید این تابع به یک جدول مسیریابی سراسری دسترسی دارد. این جدول آرایه‌ای است که سه ستون دارد و محتویات هر ستون، اعداد صحیح هستند. این ستونها عبارتند از: آدرس IP ، الگوی زیرشبکه (Subnet Mask) و شمارهٔ خط خروجی مورد استفاده [برای رسیدن به شبکه‌ای با آدرس IP متناظر]. این تابع، باید آدرس IP دریافتی از پارامتر ورودی را در جدول فوق‌الذکر به روش CIDR جستجو کرده و شمارهٔ خط خروجی متناسب با آن را به عنوان مقدار برگشتی باز گرداند.

۵۷. با استفاده از فرامین اجرایی traceroute (در یونیکس) یا tracert (در ویندوز)، مسیر رسیدن از کامپیوتر خودتان به چند دانشگاه در دیگر قاره‌ها را پیدا کنید. برخی از سایتهای متعلق به دانشگاههای معروف که می‌توانید در بررسی خود از آنها استفاده کنید، عبارتند از:

www.berkeley.edu	(کالیفرنیا)
www.mit.edu	(ماساچوست)
www.vu.nl	(آمستردام)
www.ucl.ac.uk	(لندن)
www.usyd.edu.au	(سیدنی)
www.u-tokyo.ac.jp	(توکیو)
www.uct.ac.za	(کیپ تاون)

۶ لایه انتقال

لایهٔ انتقال (Transport Layer) ، فقط در یک لایه خلاصه نمی‌شود بلکه قلب تپندهٔ سلسلهٔ پروتکلهای شبکه است. وظیفهٔ این لایه آن است که داده‌ها را به روشی قابل اعتماد و کم هزینه از ماشین مبداء به ماشین مقصد انتقال بدهد، فارغ از آن که ماهیت شبکه یا شبکه‌های فیزیکی مورد استفاده چیست. بدون لایهٔ انتقال مفهوم پروتکلهای لایه‌ای، معنای حقیقی خود را پیدا نخواهند کرد. در این فصل جزئیات لایهٔ انتقال را شامل خدماتی که عرضه می‌کند، طراحی آن، پروتکلهای مرتبط و کارآیی آنها را بررسی خواهیم کرد.

۶-۱ خدمات انتقال (The Transport Service)

در بخشهای آتی، خدماتی را که لایهٔ انتقال ارائه می‌نماید، اجمالاً بررسی کرده و به انواع سرویسهایی که به لایهٔ کاربرد عرضه می‌شود نگاهی خواهیم انداخت. برای آن که حقیقت خدمات انتقال را آشکارتر کرده باشیم دو مجموعه از «**عملکردهای اولیه**» (Primitives) برای لایهٔ انتقال تعریف کرده‌ایم: مجموعهٔ اول عملکردهای ساده و فرضی هستند که به فهم ایدهٔ اصلی کمک می‌کند. سپس مجموعه‌ای از واسطها (Interfaces) را که عموماً در اینترنت بکار گرفته می‌شوند، معرفی خواهیم کرد.

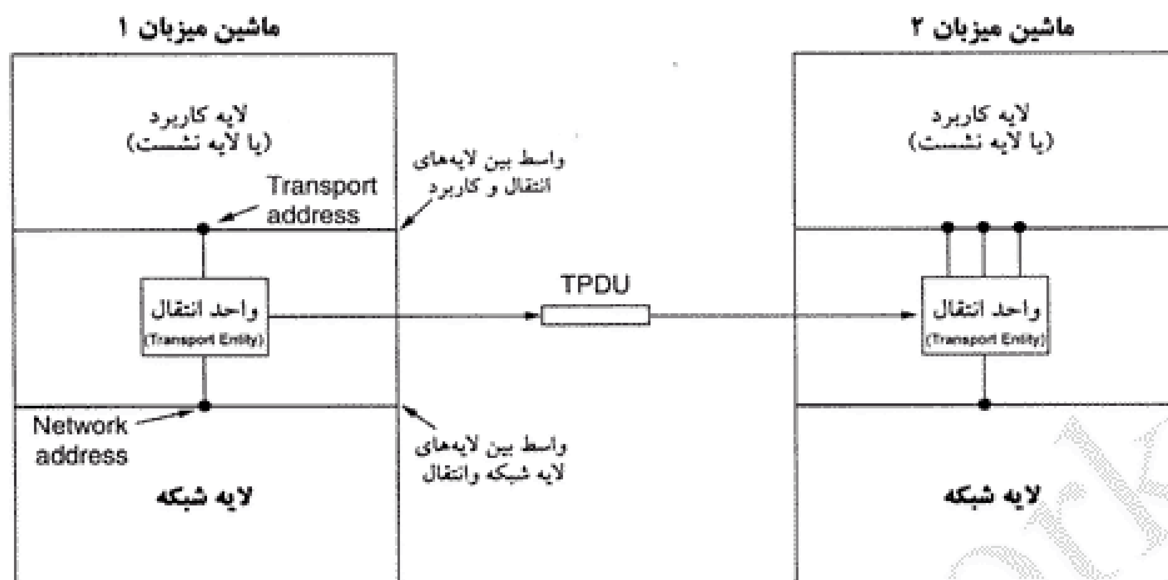
۶-۱-۱ خدمات ارائه شده به لایه‌های بالاتر

هدف نهایی لایهٔ انتقال آن است که سرویسی کارآمد، مطمئن و کم هزینه به کاربران خود بدهد. کاربران لایهٔ انتقال، پروسه‌های لایهٔ کاربرد (یا به عبارتی برنامه‌های کاربردی) هستند. برای نائل آمدن به این هدف، لایهٔ انتقال از خدماتی که لایهٔ شبکه عرضه کرده، بهره می‌گیرد. نرم‌افزار یا سخت‌افزاری که در لایهٔ انتقال این عملیات را انجام می‌دهد اصطلاحاً «**واحد انتقال**» (Transport Entity) نامیده می‌شود. واحد انتقال ممکن است درون هستهٔ سیستم عامل یا به صورت یک پروسهٔ کاربری مجزا یا در قالب یک بستهٔ کتابخانه‌ای[1] در بطن برنامه‌های کاربردی شبکه و یا حتی در درون کارت واسط شبکه تعبیه شده باشد. در شکل ۶-۱ ارتباط منطقی بین لایه‌های شبکه، انتقال و کاربرد به تصویر کشیده شده است.

همانگونه که خدمات لایهٔ شبکه بر دو نوع اتصال‌گرا و بدون‌اتصال ارائه می‌شود، خدمات لایهٔ انتقال نیز بر همین دو نوع است. خدمات مبتنی بر اتصال (هماهنگیهای قبلی) در لایهٔ انتقال از بسیاری جهات مشابه با خدمات اتصال‌گرای لایهٔ شبکه است. در هر دو مورد، «**اتصال**» سه مرحلهٔ تکوین دارد: «ایجاد اتصال» (Establishment)،

---

۱. Library Package



شکل ۶-۱. لایه‌های شبکه، انتقال و کاربرد.

«انتقال داده» (Data Transfer)، «ختم اتصال» (Release). آدرس‌دهی و کنترل جریان نیز در هر دو لایه مشابه هم هستند. مضاف بر این، خدمات بدون‌اتصال در لایهٔ انتقال نیز شباهت بسیار زیادی به خدمات بدون‌اتصال در لایهٔ شبکه دارد.

در اینجا یک سؤال بدیهی به ذهن خطور می‌کند: اگر خدمات لایهٔ انتقال این قدر به خدمات لایهٔ شبکه شبیه است، پس چرا این دو لایه از هم جدا هستند؟ چرا فقط به یکی از آنها بسنده نمی‌شود؟ پاسخ این سؤال سرراست ولی بنیادی است و از شکل ۱-۹ نشأت می‌گیرد. کد اجرایی لایهٔ انتقال کلاً بر روی ماشین کاربران اجرا می‌شود در حالی که لایهٔ شبکه اغلب بر روی مسیریابها اجرا می‌گردد که آنها نیز تحت مدیریت یک «حامل» (Carrier) هستند (حداقل در WAN اینگونه است). اگر لایهٔ شبکه خدماتی ناکافی عرضه کند (که اغلب اینگونه است) چه اتفاقی می‌افتد؟ اگر لایهٔ شبکه (به دلائلی مثل ازدحام، بروز حلقه یا پایان عمر بسته) تعدادی از بسته‌ها را از دست بدهد چه باید کرد؟ اگر یک مسیریاب هر از گاهی از کار بیفتد چه می‌شود؟

در موارد فوق فقط می‌توان گفت که مشکلات جدّی رخ می‌دهد! کاربران هیچ کنترل واقعی بر لایهٔ شبکه ندارند و طبعاً نمی‌توانند مشکلات ناشی از خدمات ناقص مسیریابها را مثلاً با تعویض مسیریاب یا مدیریت بهتر خطا در لایهٔ پیوند داده برطرف کنند. تنها راه ممکن آن است که بر روی لایهٔ شبکه، لایهٔ دیگری قرار داده شود تا کیفیت خدمات را بهبود بدهد. در یک زیر شبکهٔ اتصال‌گرا، اگر **«واحد انتقال»** (Transport Entity) در حین انتقال طولانی مدت بسته‌ها، به ناگاه متوجه شود که اتصال شبکه قطع شده و از سرنوشت داده‌های در حال انتقال نیز بی‌خبر باشد، می‌تواند یک اتصال جدید با «واحد انتقال راه دور» (Remote Transport Entity) برقرار نماید؛ به کمک اتصال جدید، واحد انتقال می‌تواند از همتای خود سؤال کند که کدام بخش از داده‌ها دریافت کرده و کدام بخش را دریافت نکرده و از جایی که داده‌ها نرسیده‌اند، شروع به ارسال مجدد نماید.

در حقیقت، وجود لایهٔ انتقال این امکان را فراهم آورده تا خدمات انتقال داده‌ها، قابل اعتمادتر از خدمات لایهٔ زیرین یعنی لایهٔ شبکه باشد. بسته‌های گمشده یا تکراری توسط لایهٔ انتقال کشف می‌شوند. مضاف بر اینها، عملکردهای اولیهٔ لایهٔ انتقال می‌تواند به صورت فراخوانی توابع کتابخانه‌ای پیاده‌سازی و در اختیار پروسه‌های لایهٔ بالاتر گذاشته شود تا لایهٔ بالاتر از عملکردهای اولیه و توابع پایهٔ لایهٔ شبکه (و درگیری با جزئیات این لایه که

امکانات ضعیفی نیز ارائه می‌کند) مستقل باشد زیرا خدماتی که لایهٔ شبکه عرضه می‌نماید در شبکه‌های متفاوت می‌تواند تفاوتهای بنیادی داشته باشد (مثلاً خدمات یک شبکهٔ بدون‌اتصال با خدمات اتصال‌گرای یک WAN تفاوت اساسی و ماهوی دارد). پنهان کردن خدمات لایهٔ شبکه در پشت یک مجموعه از «عملکردها و توابع اولیه» موجب می‌شود که در صورت تغییر در خدمات لایهٔ شبکه بتوان با عوض کردن توابع کتابخانه‌ای (به گونه‌ای که همان خدمات و عملکرد قبلی خود را در شبکهٔ جدید ارائه کنند)، این تغییر را در سطح همین لایه محدود نگاه داشت، [بدین ترتیب هیچ یک از اجزاء لایه‌های بالاتر از چنین تغییری آگاه نخواهند شد.]

با تکیه بر لایهٔ انتقال، برنامه‌نویسان نرم افزارهای کاربردی می‌توانند کد برنامهٔ خود را مبتنی بر یک مجموعهٔ استاندارد از توابع و عملکردهای اولیه بنویسند و هیچگونه نگرانی از بابت تفاوت در اجزاء و مکانیزمهای زیر شبکه یا انتقال نامطمئن و غیرقابل اعتماد نداشته باشند. اگر تمام شبکه‌های واقعی، بدون اشکال و قابل اعتماد بودند و همهٔ آنها خدمات مشابهی ارائه می‌کردند و این تضمین وجود می‌داشت که هیچگاه تغییر نکنند، آنگاه به خدمات لایهٔ انتقال نیازی نبود. ولیکن [به دلیل اختلافات بنیانی و تنوع بسیار زیاد لایه‌های زیرین] در دنیای واقعی، وجود این لایه، لایه‌های بالایی را از درگیری با جزئیات تکنولوژی، طراحی و نواقص زیر شبکه دور نگه می‌دارد.

به همین دلیل بسیاری از افراد، لایه‌های یک تا چهار را در یک دسته و لایه‌های بالاتر از ۴ را در دسته‌ای دیگر قرار می‌دهند. چهار لایهٔ اول را می‌توان **«ارائه دهندهٔ خدمات انتقال»** (Transport Service Provider) تصور کرد در حالی که بقیهٔ لایه‌های فوقانی، **«استفاده‌کنندگان از خدمات انتقال»** (Transport Service User) محسوب می‌شوند. تفکیک بین لایه‌های ارائه دهندهٔ خدمات و لایه‌های استفاده‌کنندهٔ از این خدمات، نقش لایهٔ انتقال را حساس و کلیدی کرده است چرا که این لایه در مرز بین این دو دسته قرار گرفته و باید به لایه‌های فوقانی خدمات انتقال مطمئن داده‌ها را ارائه بدهد.

۶-۱-۲ عملکردهای اولیه و توابع بنیانی لایهٔ انتقال

برای آن که امکان دسترسی به خدمات لایهٔ انتقال برای استفاده‌کنندگان این خدمات فراهم شود، این لایه باید در قالب یک **«واسط خدمات انتقال»** (Transport Service Interface) مجموعه‌ای از عملیات و توابع را در اختیار برنامه‌های کاربردی بگذارد.[1] هر رده از خدمات لایهٔ انتقال، «واسط» مختص به خود را دارد. در این بخش خدمات ساده و فرضی لایهٔ انتقال و واسط آن را بررسی می‌کنیم تا با اصول پایه آشنا شویم. سپس در بخشهای آتی، مروری بر مثالهای واقعی خواهیم داشت.

خدمات لایهٔ انتقال در عین شباهت با خدمات لایهٔ شبکه، تفاوتهای مهم دارند: تفاوت عمدهٔ آنها در این است که لایهٔ شبکه خدمات شبکهٔ واقعی و در حال استفاده را مدل می‌کند و شبکه‌های واقعی به دلایل مختلفی ممکن است بسته‌ها را از دست بدهند فلذا خدمات لایهٔ شبکه عموماً قابل اعتماد نیستند.

در عوض خدمات انتقال اتصال‌گرا در لایهٔ انتقال کاملاً قابل اعتماد است. در حالی که شبکه‌های واقعی مصون از خطا نیستند، لایهٔ انتقال بر روی این شبکهٔ نامطمئن قرار می‌گیرد و با مکانیزمهای خاصی که بدان خواهیم پرداخت، تمام این خطاها و مشکلات را جبران و برطرف می‌نماید.

به عنوان مثال، دو پروسه را در نظر بگیرید که در محیط یونیکس از طریق یک «لوله» (Pipe) به یکدیگر مرتبط شده‌اند. این دو پروسه فرض را بر آن می‌گذارند که ارتباط آنها بی‌نقص و مصون از هر نوع خطایی است. این دو تمایلی ندارند که در خصوص مکانیزمهایی مثل تصدیق دریافت داده‌ها (Acknowledgement)، بسته‌هایی که از بین می‌روند، ازدحام یا مسائلی نظیر آن چیزی بدانند. آنچه که این پروسه‌ها انتظار دارند یک ارتباط صددرصد

---

۱. مفهوم «واسط یا interface» را همان مفهوم نرم‌افزاری آن در مدل برنامه‌نویسی در نظر بگیرید. -م

مطمئن و مصون از خطاست. پروسهٔ A داده‌های خود را در انتهای «لوله» قرار می‌دهد و پروسهٔ B داده‌ها را از ابتدای این لوله بر می‌دارد. تمام آنچه که خدمات اتصال‌گرای لایهٔ انتقال باید ارائه بدهند چیزی شبیه به همین مفهوم است: یعنی پروسه‌های کاربری واقع بر هر نقطه از شبکه‌ای که نامطمئن است، بتوانند یک دنبالهٔ بیت مصون از خطا (Error Free Bit Stream) را ارسال یا دریافت کنند و نقایص زیر شبکه از چشم آنها پنهان بماند.

البته لایهٔ انتقال می‌تواند خدمات نامطمئن و بدون‌اتصال (از نوع دیتاگرام) نیز ارائه بدهد ولیکن حرف زیادی برای گفتن ندارد و تمرکز اصلی ما در این فصل بر روی خدمات اتصال‌گرا و قابل اعتماد در لایهٔ انتقال است. علیرغم آن، برخی از برنامه‌های کاربردی مثل عملیات چندرسانه‌ای، از مزایای انتقال بدون‌اتصال بهره گرفته‌اند لذا چند کلمه‌ای در خصوص آن صحبت خواهیم کرد.

تفاوت دیگر بین خدمات لایهٔ شبکه و لایهٔ انتقال آن است که استفاده‌کنندگان از خدمات آنها تفاوت بنیادی دارند. خدمات لایهٔ شبکه به **«واحد انتقال»** (Transport Entity) ارائه می‌شود. کاربران بسیار کمی هستند که بخواهند «واحد انتقال» اختصاصی برای خود بنویسند یا برنامه‌های آنها مستقیماً از خدمات حداقل و ناقص لایهٔ شبکه استفاده نماید. برعکس، اکثر برنامه‌ها (و طبعاً برنامه‌نویسان) فقط عملکردهای اولیه و توابع بنیانی لایهٔ انتقال را می‌بینند. در نتیجه، خدمات لایهٔ انتقال باید سهل‌الوصول و استفاده از آنها ساده و سرراست باشد.

برای آن که احساسی از ماهیت خدمات لایهٔ انتقال پیدا کنید به پنج عملکرد (تابع) اولیه (Primitives) که در جدول ۶-۲ فهرست شده، دقت کنید. این توابع که نقش یک «واسط انتقال» (Transport Interface) را ایفاء می‌کنند، حداقل نیازهای مورد انتظار از لایهٔ انتقال محسوب می‌شوند. این مجموعهٔ توابع، به برنامه‌های کاربردی امکان می‌دهد تا بتوانند اتصالاتی «ایجاد»، «استفاده» و نهایتاً آن را «ختم» نمایند. چنین امکانی برای اغلب برنامه‌های کاربردی کافی است.

نام عملکرد (تابع) اولیه	بسته ارسالی	توصیف
LISTEN	(none)	متوقف (بلوکه) می‌شود تا آنکه پروسه‌ای سعی در برقراری اتصال کند.
CONNECT	CONNECTION REQ.	بصورت فعال سعی در برقراری یک اتصال می‌کند.
SEND	DATA	داده می‌فرستد.
RECEIVE	(none)	متوقف می‌شود تا آنکه بسته داده برسد.
DISCONNECT	DISCONNECTION REQ.	تلاش برای قطع (خاتمه) اتصال

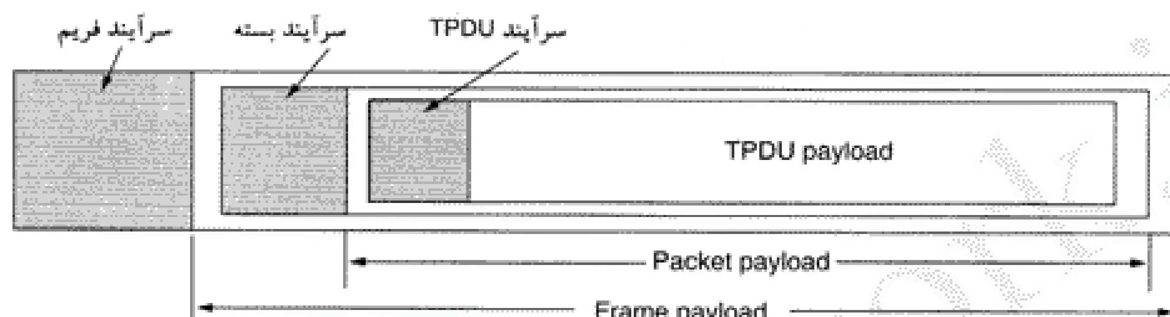
شکل ۶-۲. عملکردهای (توابع) اولیه برای ارائهٔ خدمات سادهٔ انتقال.

برای آن که ببینیم از این عملکردها و توابع چگونه استفاده می‌شود، یک برنامهٔ کاربردی شامل یک «سرویس‌دهنده» و تعدادی «مشتری» راه دور را مدّ نظر قرار بدهید. برای شروع، برنامهٔ سرویس‌دهنده با فراخوانی سیستمی، تابع LISTEN را اجرا می‌کند. اجرای این تابع سرویس‌دهنده را بلوکه کرده و به حالت انتظار می‌برد تا آن که یک «مشتری» ظاهر شود (بعبارت دیگر از راه دور تقاضای ارتباط کند). در طرف مقابل، وقتی مشتری می‌خواهد با سرویس‌دهنده محاوره کند، تابع CONNECT را اجرا می‌کند. «واحد انتقال» (Transport Entity) ضمن بلوکه کردن برنامهٔ مشتری، با ارسال بسته‌ای به سرویس‌دهنده این تقاضا را به اطلاع او می‌رساند. در درون فیلد دادهٔ این بسته، پیام لایهٔ انتقال قرار گرفته که نهایتاً تحویل «واحد انتقال» خواهد شد.

در اینجا بایستی یک واژه را معرفی نمائیم: به دلیل فقدان واژهٔ بهتر، با بی‌میلی مجبوریم برای نامگذاری ساختار پیامهایی که بین «واحدهای انتقال» مبادله می‌شوند، از واژهٔ **TPDU**[1] استفاده نمائیم. طبعاً TPDU (که بین لایه‌های

۱. Transport Protocol Data Unit

انتقال مبادله خواهند شد) در درون «بستهٔ لایهٔ شبکه» [مثلاً درون یک بستهٔ IP] جاسازی و حمل می‌شود. خود «بسته» نیز در درون یک فریم جاسازی و توسط لایهٔ پیوند داده، مبادله می‌شود. وقتی فریمی دریافت گردد، ابتدا لایهٔ پیوند داده، سرآیند آن فریم را پردازش کرده و محتوای آن را به «واحد شبکه» (Network Entity) تسلیم می‌کند. واحد شبکه نیز سرآیند بسته را پردازش کرده و محتوای فیلد دادهٔ آن بسته را تحویل «واحد انتقال» در لایهٔ بالا می‌نماید. شکل ۶-۳، تودرتویی این بسته‌ها را به تصویر کشیده است.



شکل ۶-۳. تودرتویی TPDUها، بسته‌ها و فریمها.

به مثال برنامهٔ سرویس‌دهنده/مشتری خودمان برگردیم: فراخوانی تابع CONNECT در برنامهٔ مشتری موجب می‌شود که یک بستهٔ CONNECTION REQUEST TPDU به سوی سرویس‌دهنده، ارسال شود. هر گاه این بسته دریافت شود، واحد انتقال بررسی می‌کند که آیا سرویس‌دهنده‌ای با اجرای LISTEN بلوکه شده و منتظر است؟ (به عبارتی بررسی می‌کند که آیا پروسه‌ای علاقمند به پردازش و پاسخ به چنین تقاضایی هست یا خیر)؛ اگر چنین باشد، پروسهٔ مربوطه را از حالت بلوکه خارج کرده و در پاسخ، بستهٔ CONNECTION ACCEPTED TPDU به مشتری برگردانده می‌شود. وقتی این TPDU دریافت شود، برنامهٔ مشتری نیز از حالت بلوکه خارج شده و اتصال برقرار می‌شود.

پس از این مراحل، داده‌ها می‌توانند با استفاده از توابع SEND و RECEIVE بین دو برنامه مبادله شوند. در ساده‌ترین حالت، هر یک از طرفین می‌توانند به کمک تابع RECEIVE به حالت انتظار وارد شده و منتظر بمانند تا طرف مقابل با تابع SEND اقدام به ارسال داده نماید. وقتی این TPDU دریافت شد، گیرنده از حالت بلوکه خارج می‌گردد. این پروسه نیز TPDU مربوطه را دریافت و پاسخ لازم را برمی‌گرداند. مادامی که طرفین ارتباط به نوبت ارسال نمایند این الگو به خوبی کار می‌کند.

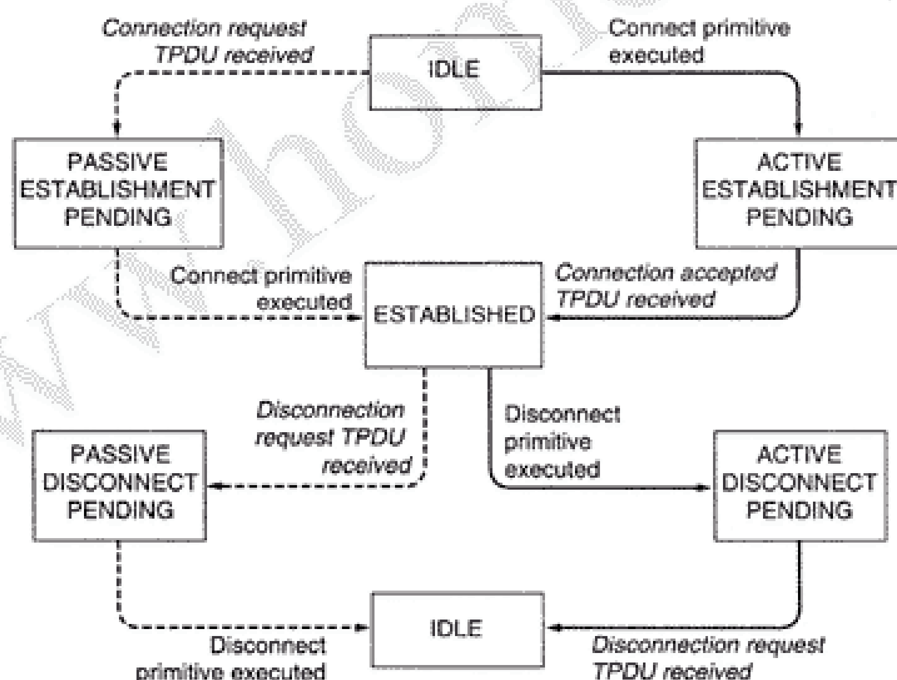
دقت کنید که لایهٔ انتقال حتی برای مبادله یکطرفه و سادهٔ داده‌ها بسیار پیچیده‌تر از لایهٔ شبکه عمل می‌کند. دریافت یکایک بسته‌هایی که ارسال می‌شوند باید به تایید طرف مقابل برسد؛ (با ارسال پیغام Ack). حتی بسته‌هایی که حامل TPDUهای کنترلی هستند نیز باید (به صورت مستقیم یا ضمنی) اعلام وصول شوند. تصدیق وصول بسته‌ها بر عهدهٔ «واحد انتقال» (Transport Entity) است و این کار از دید کاربران لایهٔ انتقال مخفی می‌ماند. بنابراین (طبق آنچه که در فصل سوم اشاره شد)، واحد انتقال باید نگران زمان‌سنجها (تایمرها) و ارسال مجدّد (Retransmission) داده‌ها باشد. هیچیک از این عملیات برای کاربران لایهٔ انتقال مشهود نیست. از دید کاربران لایهٔ انتقال، یک «اتصال» (Connection) به مثابهٔ یک «لولهٔ مطمئن انتقال بیت» (Bit Pipe) است: یکی از کاربران بیتها را در ابتدای این لوله تزریق می‌کند و به همان صورت در انتهای دیگر لوله، تحویل داده می‌شود. توانایی پنهان‌سازی پیچیدگیهای زیرین، پروتکلهای لایه‌ای را به یک ابزار قدرتمند تبدیل کرده است.

زمانی دیگر به یک اتصال نیازی نباشد باید آن را خاتمه داد تا فضای جدولی که در حافظهٔ هر کدام از

«واحدهای انتقال» به آن اختصاص داده شده، آزاد شود. قطع یک اتصال (Disconnection) به دو صورت ممکن است: نامتقارن و متقارن. در روش **«نامتقارن»** (Asymmetric) هر یک از پروسه‌های روبرو می‌توانند با صدور تابع پایهٔ DISCONNECT ، به صورت یکطرفه اقدام به ختم ارتباط نماید. صدور DISCONNECT موجب ارسال یک TPDU کنترلی خاص (DISCONNECT TPDU) به سوی «واحد انتقال» طرف مقابل می‌شود. پس از دریافت این TPDU ارتباط قطع می‌شود.

در روش **«متقارن»** (Symmetric) [چون ارتباط دوجهته -Bidirectional- است] هر یک از جهتهای انتقال به طور جداگانه بسته می‌شود: وقتی یکی از طرفین DISCONNECT می‌کند، منظورش آن است که دادهٔ دیگری برای ارسال ندارد ولیکن کماکان آمادهٔ پذیرش داده‌های شریک مقابل خود است. در این مدل، یک «اتصال»، صرفاً زمانی بطور کامل قطع می‌شود که هر دو طرف DISCONNECT نمایند.

در شکل ۶-۴ یک «دیاگرام حالت»[1] برای فرآیند ایجاد و ختم اتصال (به کمک توابع اولیه) ترسیم شده است. **«گذار»**[2] از یک **«حالت»**[3] به حالت دیگر بر اثر بروز یک **«رخداد»**[4] اتفاق می‌افتد، خواه این رخداد در اثر صدور یک تابع اولیه در پروسهٔ محلی کاربر، حادث شود و خواه در اثر ورود یک بستهٔ کنترلی (مثل Connection Request) رخ بدهد. برای سادگی فرض می‌کنیم دریافت هر TPDU بطور مجزا تصدیق (Ack) شود.[5] همچنین فرض کرده‌ایم که قطع ارتباط به صورت متقارن انجام می‌شود و پروسهٔ مشتری زودتر تقاضای قطع اتصال می‌دهد. لطفاً دقت کنید که این مدل کاملاً ساده است. بعداً در خصوص مدلهای ختم ارتباط بیشتر توضیح می‌دهیم.



شکل ۶-۴. یک دیاگرام حالت برای الگوی مدیریت سادهٔ اتصال. «گذار از یک حالت» که با قلم ایتالیک نشان داده شده در اثر ورود یک بسته حادث می‌شود. خطوط توپر توالی حالات برنامهٔ مشتری و خطوط نقطه‌چین توالی حالات برنامهٔ سرویس دهنده را نشان می‌دهند.

---

۱. State Diagram ۲. Transition ۳. State ۴. Event

۵. یعنی از روش Piggybacking (جاسازی Ack درون بسته‌های دادهٔ ارسالی) که در فصل سوم تشریح کردیم استفاده نشده باشد. ـم

۶-۱-۳ سوکتهای برکلی (Berkeley Socket)

حال اجازه بدهید به اختصار مجموعهٔ دیگری از عملکردهای اولیه و توابع بنیادی لایهٔ انتقال را که با نام «توابع سوکت» در یونیکس برکلی برای پروتکل TCP تعریف شده، بررسی نمائیم. از این توابع به طرز گسترده‌ای برای برنامه‌نویسی اینترنت استفاده می‌شود. این توابع در شکل ۶-۵ فهرست شده‌اند. در یک عبارت نادقیق، این توابع از همان مدل مثال قبلی ما پیروی می‌کنند با این تفاوت که ویژگیهای بهتر و انعطاف بیشتری دارند. فعلاً در اینجا به جزئیات ساختار TPDU در هر یک از این توابع نمی‌پردازیم و بررسی آن را تا تشریح TCP در همین فصل به تعویق می‌اندازیم.

نام عملکرد (تابع)	توصیف
SOCKET	یک نقطه ارتباط پایانی جدید ایجاد می‌کند.
BIND	به سوکت ایجاد شده یک آدرس محلی (شماره) مقید می‌کند.
LISTEN	تمایل برنامه کاربردی به پذیرش تقاضاهای اتصال را مشخص نموده و طول صف را معین می‌کند.
ACCEPT	فراخواننده را آنقدر متوقف و منتظر نگاه می‌دارد تا کسی سعی در ایجاد اتصال کند.
CONNECT	تلاش جهت ایجاد اتصال بصورت فعال
SEND	مقداری داده بر روی اتصال مشخص شده می‌فرستد.
RECEIVE	مقداری داده از اتصال مشخص‌شده می‌خواند.
CLOSE	ختم اتصال

شکل ۶-۵. توابع اولیهٔ سوکت برای TCP.

در هر برنامهٔ سرویس‌دهنده، چهار تابع پایهٔ جدول ۶-۵، به ترتیب اجرا می‌شوند. تابع SOCKET در پروسهٔ کاربردی یک نقطهٔ پایانی (End-Point) تعریف کرده و فضای حافظهٔ لازم را در واحد انتقال (Transport Entity) اختصاص می‌دهد.[۱] پارامترهای لازم برای فراخوانی این تابع عبارتند از: (۱) قالب آدرس‌دهی مورد نظر (Addressing Format)، (۲) نوع خدمات مورد انتظار (مثل استریم مطمئن از بایتها یا سرویس نامطمئن دیتاگرام) و (۳) پروتکل مورد نظر.

اگر فراخوانی تابع SOCKET موفق باشد یک «**اشاره‌گر فایل**» به عنوان مقدار برگشتی بازگردانده می‌شود تا در فراخوانیهای بعدی مورد استفاده قرار گیرد؛ دقیقاً مشابه با همان کاری که تابع OPEN برای باز کردن یک فایل انجام می‌دهد.

سوکتهایی که ایجاد می‌شوند دارای آدرسهای شبکه نیستند.[۲] برای انتساب آدرسها به هر سوکت از تابع پایهٔ BIND استفاده می‌شود. به محض آنکه سرویس‌دهنده، آدرسی را به یک سوکت نسبت داد هر مشتری راه دور می‌تواند اقدام به برقراری اتصال با آن کند.[۳] دلیل آن که با فراخوانی تابع SOCKET ، آدرس لازم به آن انتساب داده نمی‌شود و تابعی مجزا برای آن در نظر گرفته شده، آن است که برخی از پروسه‌ها در مورد آدرس مورد نظر خود مطمئن هستند (مثلاً برای سالها از یک آدرس استفاده می‌نمایند و همهٔ مشتریان از این شماره آگاهند) در حالی که برای برخی دیگر از پروسه‌ها، مقدار این آدرس و ثبات آن اصلاً اهمیت ندارد.

---

۱. در حقیقت با تابع SOCKET هویت هر یک از طرفین در دو سر خط لولهٔ انتقال مشخص می‌شود. -م

۲. به عبارت دیگر، هر چند یک نقطهٔ پایانی در پروسه ایجاد می‌شود ولی هیچ آدرسی که بتوان از راه دور با آن اقدام به برقراری تماس کرد وجود ندارد. -م

۳. به فرآیند انتساب آدرس به یک سوکت عمل «مقیدسازی» سوکت - Binding - گفته می‌شود. -م

در ادامه، پروسهٔ سرویس‌دهنده تابع پایهٔ LISTEN را فراخوانی می‌کند. اجرای این تابع فضای لازم را برای صف‌بندی تقاضاهای ایجاد اتصال، اختصاص می‌دهد. بدین ترتیب سرویس‌دهنده می‌تواند بطور همزمان با چندین مشتری اتصال ایجاد کند. برخلاف مدل مثال اول، در اینجا فراخوانی تابع LISTEN پروسهٔ سرویس‌دهنده را بلوکه و معلق نخواهد کرد.

برای آن که پروسه تا دریافت تقاضای ایجاد اتصال، معلق و منتظر بماند سرویس‌دهنده، تابع ACCEPT را فراخوانی می‌کند. هر گاه یک TPDU مبنی بر تقاضای ایجاد یک اتصال دریافت شود، واحد انتقال سوکت جدیدی با همان مشخصات سوکت اصلی ایجاد کرده و اشاره‌گر آن را بر می‌گرداند. برنامهٔ سرویس‌دهنده می‌تواند یک پروسهٔ فرزند (Child Process) یا یک «ریسمان» (Tread) برای آن ایجاد کند تا به سرویس‌دهی به اتصال جدید مشغول شود. سپس مجدداً به حالت انتظار برگشته تا بتواند تقاضاهای اتصال جدید را بپذیرد. تابع ACCEPT یک اشاره‌گر معمولی برمی‌گرداند تا با استفاده از آن بتوان به روش معمولی داده‌ای خواند یا نوشت. (دقیقاً شبیه به عملیات خواندن و نوشتن از فایل که با اشاره‌گر مربوطه انجام می‌شود.)

حال اجازه بدهید نگاهی به پروسهٔ سمت مشتری بیندازیم: در اینجا نیز بایستی با فراخوانی تابع پایهٔ SOCKET ، یک سوکت ایجاد شود ولیکن نیازی به فراخوانی تابع BIND نیست چرا که در اینجا آدرس پروسهٔ مشتری برای سرویس‌دهنده اهمیتی ندارد.[۱] سپس با فراخوانی تابع CONNECT ، پروسهٔ فراخواننده معلق شده و همان وقت فرآیند ایجاد اتصال آغاز می‌شود. وقتی این فرآیند تکمیل شود (یعنی وقتی پاسخ مناسب از سرویس‌دهنده دریافت گردد) پروسهٔ مشتری از حالت تعلیق به در آمده و اتصال مورد نظر ایجاد می‌شود. در این نقطه هر یک از طرفین این اتصال می‌توانند با فراخوانی توابع پایهٔ SEND و RECV اقدام به ارسال یا دریافت دوطرفه و همزمان (Full Duplex) داده‌ها بنمایند. در یونیکس حتی می‌توان از فراخوانیهای سیستمی READ و WRITE به جای SEND و RECV استفاده کرد.

مدل قطع ارتباط در سوکتهای فوق، متقارن است یعنی زمانی که هر دو طرف، تابع CLOSE را اجرا کنند، اتصال ایجاد شده خاتمه خواهد یافت.

۶-۱-۴ مثالی از برنامه‌نویسی سوکت: یک سرویس‌دهندهٔ اینترنتی فایل

به عنوان مثالی از چگونگی فراخوانی توابع سوکت، کدهای برنامهٔ سرویس‌دهنده و مشتری را در شکل ۶-۶ مدنظر قرار بدهید. در این مثال یک سرویس‌دهندهٔ ابتدایی فایل به همراه یک برنامهٔ مشتری (که از آن بهره می‌گیرد) ، ارائه شده است. این قطعه کُد از محدودیتهای بی‌شماری که در زیر تشریح کرده‌ایم، رنج می‌برد ولیکن در هر حال می‌توان کد برنامهٔ سرویس‌دهنده را کامپایل کرده و آن را بر روی هر سیستم یونیکس متصل به اینترنت اجرا کرد. کد برنامهٔ مشتری را نیز می‌توان پس از کامپایل، بر روی هر ماشین یونیکس متصل به اینترنت در سراسر دنیا اجرا و از آن استفاده کرد. برنامهٔ مشتری در خط فرمان (به همراه دو آرگومان) اجرا می‌شود تا بکمک آن هر فایلی را که برنامهٔ سرویس‌دهنده بدانها دسترسی دارد، دریافت کرد. فایل دریافتی بر روی «خروجی استاندارد» نوشته می‌شود و بدیهی است که می‌توان به کمک مفهوم **«تغییر مسیر»** (Redirection) در یونیکس، آنرا به درون یک فایل یا یک «لوله» (Pipe) هدایت کرد.

اجازه بدهید به کد برنامهٔ سرویس‌دهنده نگاهی بیندازیم. این برنامه با تعریف چند include file[۲] شروع می‌شود. سه فایل آخر، در برگیرنده تعاریف مرتبط با اینترنت و ساختمان دادهٔ مورد نیاز شبکه است. سپس ثابت

---

۱. دقت کنید که پروسهٔ مشتری قطعاً دارای آدرس هست ولی فقط مقدار آن مهم نیست و در اینجا انتخاب آن بر عهدهٔ لایهٔ انتقال گذاشته شده است. -م

۲. مفهوم include# را در زبان C ، در نظر داشته باشید.

SERVER-PORT معادل با 12345 تعریف شده است. این عدد کاملاً اختیاری است: اعداد بین ۱۰۲۴ تا ۶۵۵۳۵ (به شرط آن که توسط پروسهٔ دیگری استفاده نشده باشد) قابل انتخاب است. البته برنامه سرویس‌دهنده و مشتری باید از شمارهٔ مشابهی استفاده کنند. اگر روزی این سرویس‌دهنده جهانی شود باید به آن یک شمارهٔ پورت زیر ۱۰۲۴ انتساب داده شود و در سایت www.iana.com به اطلاع همه برسد!!

دو خط بعدی از برنامهٔ سرویس‌دهنده، دو ثابت مورد نیاز برنامه را تعریف کرده است. اولین آنها حداکثر طول قطعات ارسالی فایل را مشخص نموده است. دومین ثابت، تعیین می‌کند که حداکثر چند اتصال معلق و منتظر را می‌توان حفظ کرد.

پس از تعریف متغیرهای محلی، کد برنامهٔ سرویس‌دهنده آغاز می‌شود. در ابتدا یک ساختمان داده با آدرس IP ماشین سرویس‌دهنده، مقداردهی اولیه می‌شود. در ادامه، این ساختمان داده به سوکت سرویس‌دهنده، «مقید» (Bind) خواهد شد. فراخوانی تابع *memset* کلّ ساختمان داده را با صفر پر می‌کند. سپس سه دستور انتساب بعدی فیلدهای این ساختمان داده را مقداردهی می‌نمایند. آخرین دستور انتساب، شمارهٔ پورت سرویس‌دهنده را مشخص کرده است. توابع htonl و htons مقادیر فیلدها را به قالب استاندارد تبدیل می‌کند تا این برنامه بتواند بدرستی بر روی ماشینهای Big Endian (مثل ماشینهای SPARC) و ماشینهای Little Endian (مثل ماشینهای پنتیوم) اجرا شود.[1]

در ادامه، برنامهٔ سرویس‌دهنده سوکتی را ایجاد کرده و خطای احتمالی را بررسی می‌نماید. (هر گونه خطا با $s<0$ مشخص می‌شود.) در نسخهٔ واقعی این برنامه، پیامهای خطا می‌توانند گویاتر و مفصل‌تر باشند. فراخوانی تابع *setsockopt* بدان جهت نیاز است که بتوان از آن پورت به دفعات استفاده کرد و برنامهٔ سرویس‌دهنده بتواند به صورت نامحدود اجرا شود. حال به سوکت ایجاد شده آدرس IP و پورت، مقید (Bind) شده و بررسی می‌شود که آیا این عمل موفق بوده است. آخرین گام در مقداردهی اولیه، فراخوانی تابع *listen* است تا تمایل سرویس‌دهنده به پذیرش تقاضاهای اتصال را به اطلاع سیستم رسانده و مشخص کند که سیستم فقط حق دارد به تعداد QUEUE_SIZE از متقاضیان اتصال را (در حین پردازش یکی از آنها) پذیرفته و معلق نگه دارد. اگر صف مربوطه پر باشد و تقاضای اتصال جدیدی برسد، نادیده گرفته می‌شود.

در این نقطه، برنامهٔ سرویس‌دهنده به حلقهٔ اصلی خود وارد می‌شود؛ حلقه‌ای که هرگز از آن خارج نخواهد شد. تنها راه متوقف کردن این برنامه، «کُشتن» آن از بیرون (با فرمان kill) است. فراخوانی تابع accept، سرویس‌دهنده را بلوکه می‌کند تا آنکه یک یا چند مشتری سعی کنند با آن اتصالی را برقرار نمایند. اگر فراخوانی تابع accept موفق باشد، یک اشاره‌گر معمولی فایل، بازگردانده می‌شود تا به کمک آن بتوان داده ارسال یا دریافت کرد. (به همان نحوی که با اشاره‌گر فایل می‌توان درون یک لوله (Pipe) نوشت یا از آن خواند.) ولیکن برخلاف «لوله» در یونیکس که یک طرفه است، سوکتها دوطرفه هستند لذا با در اختیار داشتن آدرس سوکت (یعنی متغیر *sa*) می‌توان بر روی اتصال ایجاد شده نوشت (معادل ارسال) یا از آن خواند (معادل دریافت).

پس از آن که اتصال ایجاد شد، برنامهٔ سرویس‌دهنده نام فایل مورد نظر مشتری را از روی آن اتصال می‌خواند. اگر داده‌ای دریافت نشده باشد، سرویس‌دهنده بلوکه شده و منتظر می‌ماند. پس از دریافت نام فایل مورد نظر مشتری، سرویس‌دهنده، فایل مربوطه را باز کرده و در حلقهٔ دیگری وارد می‌شود تا متوالیاً بلوکهای فایل را خوانده و آن را بر روی سوکت ارسال نماید. این حلقه آنقدر ادامه می‌یابد تا کل فایل منتقل شود. سپس فایل و اتصال متناظر را می‌بندد و منتظر اتصال بعدی می‌ماند. این حلقه تا ابد ادامه دارد.

---

۱. برای آشنایی با تعریف این واژه‌ها به فصول قبلی مراجعه کنید.

```
/* This page contains a client program that can request a file from the server program
 * on the next page. The server responds by sending the whole file.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345              /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                  /* block transfer size */

int main(int argc, char **argv)
{
 int c, s, bytes;
 char buf[BUF_SIZE];                   /* buffer for incoming file */
 struct hostent *h;                    /* info about server */
 struct sockaddr_in channel;           /* holds IP address */

 if (argc != 3) fatal("Usage: client server-name file-name");
 h = gethostbyname(argv[1]);           /* look up host's IP address */
 if (!h) fatal("gethostbyname failed");

 s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (s <0) fatal("socket");
 memset(&channel, 0, sizeof(channel));
 channel.sin_family= AF_INET;
 memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
 channel.sin_port= htons(SERVER_PORT);

 c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
 if (c < 0) fatal("connect failed");

 /* Connection is now established. Send file name including 0 byte at end. */
 write(s, argv[2], strlen(argv[2])+1);

 /* Go get the file and write it to standard output. */
 while (1) {
      bytes = read(s, buf, BUF_SIZE);  /* read from socket */
      if (bytes <= 0) exit(0);         /* check for end of file */
      write(1, buf, bytes);            /* write to standard output */
 }
}

fatal(char *string)
{
 printf("%s\n", string);
 exit(1);
}
```

شکل ۶-۶. کُد برنامهٔ مشتری با بهره‌گیری از سوکتها. کُد برنامهٔ سرویس‌دهنده در صفحهٔ بعدی آمده است.

```
#include <sys/types.h>                      /* This is the server code */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                   /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                       /* block transfer size */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                       /* buffer for outgoing file */
  struct sockaddr_in channel;               /* holds IP address */

  /* Build address structure to bind to socket. */
  memset(&channel, 0, sizeof(channel));     /* zero channel */
  channel.sin_family = AF_INET;
  channel.sin_addr.s_addr = htonl(INADDR_ANY);
  channel.sin_port = htons(SERVER_PORT);

  /* Passive open. Wait for connection. */
  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);    /* create socket */
  if (s < 0) fatal("socket failed");
  setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
  if (b < 0) fatal("bind failed");

  l = listen(s, QUEUE_SIZE);                /* specify queue size */
  if (l < 0) fatal("listen failed");

  /* Socket is now set up and bound. Wait for connection and process it. */
  while (1) {
       sa = accept(s, 0, 0);                /* block for connection request */
       if (sa < 0) fatal("accept failed");

       read(sa, buf, BUF_SIZE);             /* read file name from socket */

       /* Get and return the file. */
       fd = open(buf, O_RDONLY);            /* open the file to be sent back */
       if (fd < 0) fatal("open failed");

       while (1) {
            bytes = read(fd, buf, BUF_SIZE); /* read from file */
            if (bytes <= 0) break;          /* check for end of file */
            write(sa, buf, bytes);          /* write bytes to socket */
       }
       close(fd);                           /* close file */
       close(sa);                           /* close connection */
  }
}
```

حال اجازه بدهید به کد برنامهٔ مشتری نگاهی بیندازیم. برای آشنایی با عملکرد این برنامه، لازم است با روش فراخوانی و اجرای آن در خط فرمان، آشنا شوید. با فرض آن که نام این برنامه client انتخاب شده باشد، فراخوانی آن همانند زیر است:

```
client flits.cs.vu.nl /usr/tom/filename >f
```

فراخوانی برنامهٔ فوق زمانی موفق است که برنامهٔ سرویس‌دهنده از قبل بر روی ماشین flits.cs.vu.nl اجرا شده و همچنین فایلی با مشخصات usr/tom/filename/ موجود و دسترسی برنامهٔ سرویس‌دهنده به آن مجاز باشد. اگر فراخوانی برنامه با موفقیت انجام شود، فایل فوق‌الذکر از طریق اینترنت منتقل و درون **f** نوشته می‌شود؛ پس از آن برنامهٔ client به پایان می‌رسد. از آنجایی که اجرای برنامهٔ سرویس‌دهنده پس از انتقال یک فایل ادامه خواهد یافت فلذا برنامهٔ مشتری را می‌توان بارها اجرا و فایلهای دیگری را دریافت کرد.

در ابتدای برنامهٔ مشتری نیز برخی از تعاریف اولیه آمده است. اجرای واقعی برنامه با بررسی تعداد آرگومانها آغاز می‌شود. (*argc*=3 به معنای سه آرگومان، شامل نام برنامه بهمراه دو آرگومان دیگر است.) دقت کنید که *argv*[1] حاوی نام ماشین سرویس‌دهنده (مثل flits.cs.vu.nl) است و بعداً به کمک تابع سیستمی *gethostbyname* به آدرس IP ترجمه و تبدیل خواهد شد. این تابع سیستمی برای تبدیل نام از سیستم DNS بهره می‌گیرد. DNS را در فصل هفتم تشریح خواهیم کرد.

در ادامه یک سوکت ایجاد و مقداردهی اولیه می‌شود. سپس برنامهٔ مشتری تلاش می‌کند با استفاده از تابع *connect* یک اتصال TCP با سرویس‌دهنده برقرار نماید. اگر سرویس‌دهنده بر روی ماشین با نام مورد نظر، اجرا شده باشد و همچنین به SERVER_PORT گوش بدهد و فضای کافی در صف *listen* وجود داشته باشد، یک اتصال برقرار می‌شود. از طریق این اتصال، برنامهٔ مشتری نام فایل مورد نظر خود را برای سرویس‌دهنده ارسال می‌کند. این کار با تابع *write* انجام می‌شود. پارامتر آخر در تابع *write*، تعداد بایتهایی را که باید ارسال شوند، مشخص می‌کند؛ از آنجایی که نام فایل به همراه کاراکتر '\0' ارسال می‌شود لذا تعداد بایتها یکی بیشتر از طول واقعی رشتهٔ حاوی نام در نظر گرفته شده است.

در اینجا برنامهٔ مشتری به یک حلقه وارد شده و فایل را بلوک به بلوک از سوکت خوانده و در «خروجی استاندارد» کپی می‌نماید. پس از این کار، برنامه به اتمام می‌رسد.

عملکرد زیربرنامهٔ fatal (در صورت فراخوانی) آنست که، یک پیام خطا بر روی خروجی چاپ کرده و از برنامه خارج شود. دقت کنید که برنامهٔ سرویس‌دهنده نیز به همین زیربرنامه احتیاج دارد ولی به دلیل کمبود جا در صفحه، از تعریف آن چشمپوشی شده است. از آنجایی که برنامهٔ سرویس‌دهنده و برنامه مشتری بطور جداگانه کامپایل و بر روی کامپیوترهای متفاوتی اجرا می‌شوند لذا نمی‌توانند از کُد زیربرنامهٔ fatal به صورت اشتراکی بهره بگیرند. این دو برنامه را (بهمراه برخی دیگر از مفاد مرتبط با کتاب) می‌توانید از وب‌سایت *http://www.prenhall.com/tanenbaum* بدست بیاورید. در این سایت بر روی تصویر روی جلد کتاب کلیک کنید؛ در صفحه‌ای که ظاهر می‌شود می‌توانید برنامه‌های فوق را دریافت و آن را بر روی هر سیستم سازگار با یونیکس (مثل سولاریس، BSD و لینوکس) به نحو زیر کامپایل کنید:

```
cc -o client client.c -lsocket -lnst
cc -o server server.c -lsocket -lnst
```

برنامهٔ سرویس‌دهنده با درج نام برنامه بر روی خط فرمان اجرا می‌شود:

```
server
```

به نحوی که اشاره شد برنامه مشتری به دو آرگومان نیاز دارد. نسخهٔ Windows این دو برنامه نیز در وب‌سایت

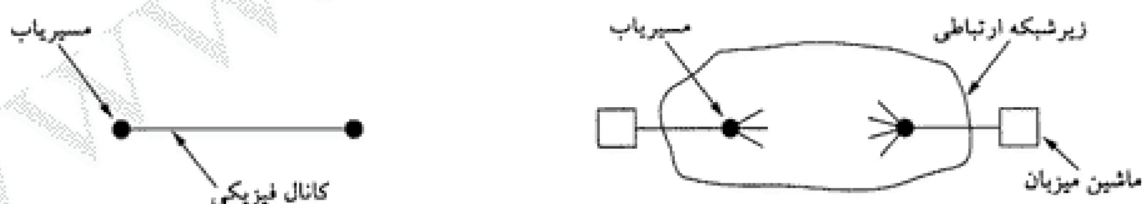
فوق در دسترس می‌باشد.

یادآوری می‌کنیم که این سرویس‌دهنده، از بسیاری جهات تکامل یافته نیست. پردازش و مدیریت خطای آن بسیار ضعیف و گزارشهای خطا ناچیز است. هیچ تمهیدی در خصوص امنیت در آن اندیشیده نشده و فراخوانیهای سیستمی یونیکس در آن، «ویژگی استقلال از محیط اجرا»[1] را مخدوش کرده است. در ضمن این برنامه براساس فرضیاتی نوشته شده است که اصولاً اشتباه هستند؛ به عنوان مثال فرض شده که نام فایل در بافر جا می‌گیرد و به صورت یکجا ارسال می‌شود. از آنجایی که این برنامه تمام تقاضاها را صرفاً به صورت ترتیبی و پشت سرهم پاسخ می‌دهد (چرا که فقط یک «ریسمان» ـThreadـ دارد)، فلذا کارآیی آن بشدت پایین است. با تمام این کاستی‌ها، برنامهٔ فوق کامل بوده و می‌تواند به صورت یک سرویس‌دهندهٔ فایل ایفای نقش کند. خوانندگان گرامی را به توسعهٔ این برنامه دعوت می‌نماییم. برای کسب آگاهی بیشتر در خصوص برنامه‌نویسی سوکت به مرجع (Stevens,1997) مراجعه نمائید.

۶ـ۲ مؤلفه‌های هر پروتکل انتقال

خدمات انتقال توسط «پروتکلهای انتقال» پیاده‌سازی شده و از آنها بین دو «واحد انتقال» (Transport Entity) استفاده می‌شود. از بسیاری جهات، پروتکلهای انتقال با پروتکلهای لایهٔ پیوند داده که در فصل سوم تشریح شد، شبیه هستند. هر دو با مسئلهٔ نظارت بر خطا، حفظ ترتیب داده‌ها و کنترل جریان و مواردی از این قبیل، سر و کار دارند.

در عین حال، تفاوتهای چشمگیری بین این دو وجود دارد. این تفاوتها عمدتاً ناشی از محیط و بستری است که این دو پروتکل در آن کار می‌کنند. به شکل ۶-۷ دقت کنید: در لایهٔ پیوند داده دو مسیریاب از طریق یک کانال فیزیکی مستقیماً با یکدیگر به تبادل داده می‌پردازند در حالی که در لایه انتقال این کانال فیزیکی جای خود را به یک زیر شبکهٔ کامل [با تعدادی مسیریاب و کانال مخابراتی] می‌دهد. این تفاوت مستلزم پیش‌بینی تمهیدات بسیار مهمی است که در این فصل آنها را بررسی خواهیم کرد.



شکل ۶-۷. (الف) محیط لایهٔ پیوند داده (ب) محیط لایهٔ انتقال.

اولین تفاوت آن است که در لایهٔ پیوند داده، مسیریاب نیاز ندارد که بداند با کدام مسیریاب در حال محاوره است، چراکه هر یک از خطوط خروجی به صورت یکتا، صرفاً یک مسیریاب خاص را مشخص می‌کنند.[2] برعکس در لایهٔ انتقال، نیاز است که آدرس دقیق مقصد مشخص باشد.

---

۱. Platform Independency

۲. به عبارتی دیگر، چون خطوط بین دو مسیریاب مستقیم و نقطه‌به‌نقطه هستند، می‌توان با آن مستقیماً و بدون آن که آدرس طرف مقابل مشخص باشد، مبادله داده کرد. ـم

مورد اختلاف دیگر آن است که فرآیند ایجاد یک «اتصال» بر روی سیم (نشان داده شده در شکل ۶-۷-الف) ساده است: طرف مقابل همیشه هست (مگر آنکه از کار بیفتد یا رخدادی جدی حادث شود که در این صورت هیچ کاری نمی‌توان انجام داد!) در لایهٔ انتقال به نحوی که خواهیم دید، ایجاد اتصال فرآیندی پیچیده است.

یکی دیگر از تفاوتهای بسیار آزاردهنده بین لایهٔ پیوند داده و لایهٔ انتقال آن است که زیر شبکه مستعد نگهداری و معطل کردن بسته‌ها در حافظه است. وقتی یک مسیریاب فریمی را بر روی لینک خود ارسال می‌کند بالاخره یا می‌رسد یا به دلیل خطا از دست می‌رود ولی این فریم قادر نیست مدتی سرگردان باشد، در گوشه‌ای از دنیا کور و گم شود و سی ثانیه بعد به ناگاه در مقصد ظاهر گردد. اگر زیرشبکه‌ای از روش دیتاگرام و مسیریابی پویا بهره گرفته باشد، احتمال آن که بسته‌ای در آن ذخیره و چندین ثانیه بعد تحویل مقصد شود قابل چشمپوشی نیست. تبعات ناشی از استعداد زیر شبکه در ذخیره و تعلیق بسته‌ها، بسیار مشکل‌آفرین و برای رفع آنها به پروتکلهای خاص نیازمند است.

تفاوت نهایی بین لایهٔ پیوند داده و لایهٔ انتقال بیشتر در **میزان انتظارات** است تا آن که در **نوع انتظارات** باشد: بافرسازی و کنترل جریان در هر دو لایه الزامی است ولیکن وجود تعداد بسیار زیاد و متغیر «اتصال» (Connection) در لایهٔ انتقال به راهکارها و مکانیزمهای متفاوتی (در مقایسه با راهکارهای اتخاذ شده در لایهٔ پیوند داده) احتیاج دارد. در فصل سوم دیدیم که برخی از پروتکلهای لایهٔ پیوند داده تعداد ثابت و مشخصی بافر برای هر خط اختصاص می‌دهند و هر وقت که فریمی از راه برسد، بافر مورد نیاز موجود خواهد بود. ولیکن در لایهٔ انتقال، ایدهٔ اختصاص تعداد ثابتی بافر برای هر اتصال (به دلیل آن که تعداد اتصالات می‌تواند بسیار زیاد و متغیر باشد) چندان عملی نیست. در بخشهای آتی کلیهٔ این موارد و مواردی از این قبیل بحث و بررسی خواهند شد.

۶-۲-۱ آدرس‌دهی

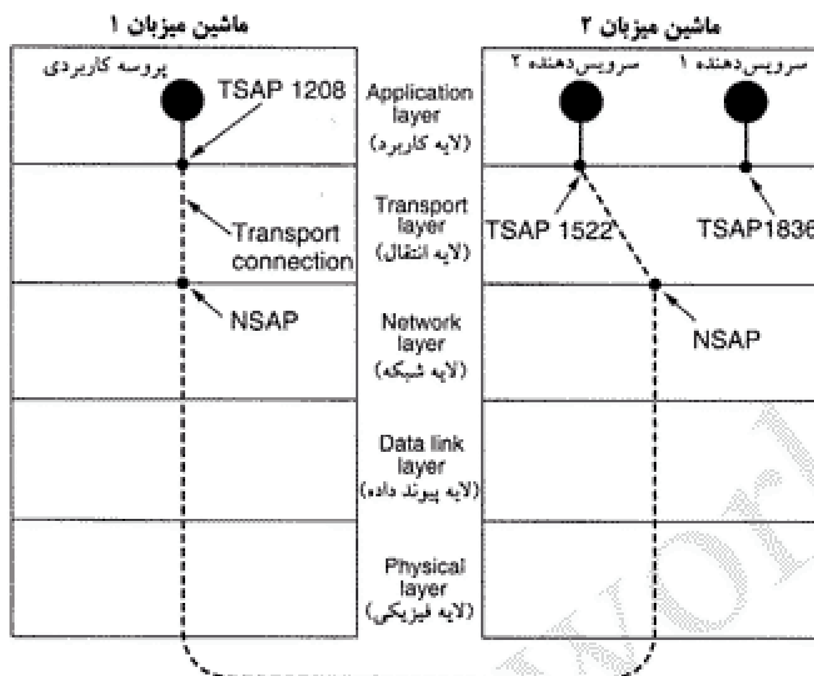
وقتی یک پروسهٔ کاربردی (برنامهٔ کاربر) می‌خواهد با یک پروسهٔ کاربردی راه دور، «اتصال» ایجاد کند باید پروسهٔ مورد نظرش را دقیقاً مشخص نماید. (انتقال بدون‌اتصال نیز با همین مسئله مواجه است: هر پیام برای چه پروسه‌ای ارسال می‌شود؟) روشی که عموماً مورد استفاده قرار می‌گیرد تعریف «آدرس انتقال» (Transport Address) برای پروسه‌هایی است که برای دریافت تقاضای اتصال در حال انتظار (شنود ـ Listening) هستند. [یعنی پروسه‌های در حال شنود توسط یک آدرس یکتا، مشخص می‌شوند.] در اینترنت، به این نقاط پایانی «**پورت**» گفته می‌شود. در شبکهٔ ATM این نقاط، AAL_SAP نام گرفته‌اند. برای آن که عمومیت مطالب تحت تأثیر این واژه‌ها قرار نگیرد ما از واژهٔ کلّی **TSAP**[۱] استفاده خواهیم کرد. (مشابه با نقاط پایانی در لایهٔ شبکه -یعنی آدرسهای لایهٔ شبکه- که اصطلاحاً NSAP نامیده می‌شد.)

شکل ۶-۸ ارتباط بین NSAP ، TSAP و اتصالات لایهٔ انتقال را به تصویر کشیده است. پروسه‌های کاربردی اعم از سرویس‌دهنده یا مشتری می‌توانند خود را به یک TSAP متصل نموده تا بتوانند با یک «TSAP راه دور»[۲] تماس (اتصال) برقرار کنند. به گونه‌ای که در این شکل نشان داده شده هر اتصال از طریق NSAP برقرار می‌شود.[۳] هدف از تعریف TSAP آن است که در شبکه‌ای عظیم که هر کامپیوتر متصل به آن عموماً با یک NSAP واحد شناسایی می‌شود، بتوان نقاط پایانی متعددی را که همگی یک NSAP مشترک دارند، از هم تمیز داد.

---

۱. Transport Service Access Point　　۲. Remote TSAP

۳. NSAP یک ماشین واحد را در کلّ شبکه و TSAP یک پروسه را بر روی آن ماشین مشخص می‌کند. ترکیب «NSAP : TSAP» به پروسه‌ها هویت جهانی و منحصر به فرد می‌دهد. -م



شکل ۶-۸ مفهوم TSAP، NSAP و اتصال.

سناریویی برای ایجاد یک اتصال در لایهٔ انتقال، می‌تواند به ترتیب زیر باشد:

۱. یک پروسهٔ سرویس‌دهنده (مثلاً پروسهٔ سرویس‌دهندهٔ Time of Day که تاریخ و ساعت را اعلام می‌کند) بر روی ماشین ۲ خودش را به TSAP 1522 متصل کرده و منتظر دریافت تقاضای تماس دیگران می‌ماند. چگونگی وصل شدن یک پروسه به یک TSAP صرفاً به سیستم عامل محلی هر ماشین بستگی دارد و در محدودهٔ مدل استاندارد شبکه قرار نمی‌گیرد. به عنوان مثال می‌توان تابع سیستمی LISTEN را فراخوانی کرد.

۲. پروسهٔ کاربردی بر روی ماشین ۱ می‌خواهد زمان و تاریخ روز را بداند لذا تقاضای CONNECT را صادر کرده و در آن هویت مبداء (یعنی خودش) را با TSAP 1028 و هویت مقصد (یعنی سرویس‌دهنده) را با TSAP 1522 مشخص می‌کند. این کار موجب می‌شود که یک اتصال بین پروسهٔ کاربردی روی ماشین ۱ با پروسهٔ سرویس‌دهندهٔ روی ماشین ۲ ایجاد شود.

۳. در اینجا پروسهٔ کاربردی تقاضای «زمان و تاریخ» را ارسال می‌کند.

۴. در پاسخ پروسهٔ سرویس‌دهنده ، زمان فعلی را اعلام می‌کند.

۵. اتصال (تماس) قطع می‌شود.

دقت کنید که ممکن است بر روی ماشین میزبان ۲ ، سرویس‌دهنده‌های دیگری نیز اجرا و به TSAP متفاوتی متصل شده باشند؛ همهٔ این سرویس‌دهنده‌ها NSAP مشابهی دارند.[۱]

تصویری که ارائه شد مفید و گویاست ولیکن نکتهٔ کوچکی را از قلم انداخته‌ایم: پروسهٔ کاربری ۱ از کجا بداند که سرویس‌دهندهٔ زمان (Time-of-Day Server) به TSAP 1522 متصل شده است؟ یک راه آن است که این سرویس‌دهنده، سالها از TSAP 1522 استفاده کرده باشد و کاربران شبکه به تدریج از این شماره آگاه شده باشند. در این رویکرد ، سرویس‌دهنده‌ها TSAP ثابت و پایداری خواهند داشت و می‌توان آنها را درون فایلی در یک

۱. در اینترنت NSAP همان آدرس IP است. -م

محل شناخته شده مثل فایل */etc/services* ذخیره کرد. (در این فایل، فهرست تمام سرویس‌دهنده‌های استاندارد و مشهور بهمراه شمارهٔ پورتهایی که بدان متصل هستند، درج شده است.)

آدرسهای ثابت و پایدار TSAP فقط برای سرویس‌دهنده‌های مهم و کلیدی (مثل سرویس‌دهندهٔ وب) مفید خواهد بود، در حالی که پروسه‌های کاربری عموماً می‌خواهند با پروسه‌هائی محاوره کنند که به صورت موقتی اجرا شده‌اند و هیچ آدرس TSAP که از قبل مشخص باشد، ندارند. مضاف بر این، سرویس‌دهنده‌های متنوعی وجود دارند که اغلب آنها کاربرد چندانی ندارند و به صورت موردی اجرا می‌شوند، لذا اختصاص آدرس TSAP ثابت به آنها و فعال نگه داشتنشان بی‌فایده است. کوتاه سخن آن که به روش بهتری نیاز است!

یک روش مناسب در شکل ۶-۹ (به صورت ساده و خلاصه) به تصویر کشیده شده است. این روش به نام **«پروتکل اتصال اولیه»** (Initial Connection Protocol) مشهور است. به جای آن که هر سرویس‌دهنده به یک TSAP شناخته شده گوش بدهد، ماشینی که علاقمند به سرویس‌دهی به کاربران راه دور است پروسهٔ خاصی به نام Process Server را اجرا می‌کند. Process Server ، وکیل تمام سرویس‌دهنده‌هایی است که کمتر مورد استفاده قرار می‌گیرند. این پروسه بطور همزمان به مجموعه‌ای از پورتها گوش داده و منتظر تقاضای برقراری ارتباط می‌ماند. کاربران احتمالی، با ارسال تقاضای CONNECT، شمارهٔ TSAP سرویس‌دهندهٔ مورد نظرشان را تعیین می‌کنند. اگر هیچ پروسه‌ای به TSAP مورد نظر آنها متصل نشده و در حال شنود نباشد، لاجرم اتصال آنها مطابق با شکل ۶-۹-الف با Process Server برقرار می‌شود.

وقتی تقاضای برقراری اتصال دریافت شد، Process Server ضمن تشخیص سرویس‌دهندهٔ مورد تقاضا آن را راه‌اندازی و اجرا می‌کند و اتصال ایجاد شده بین خودش و کاربر را به پروسهٔ سرویس‌دهنده تحویل می‌دهد. در این لحظه سرویس‌دهندهٔ تازه اجرا شده، مشغول انجام عملیات درخواستی می‌شود در حالی که Process Server به سر کار اصلیش یعنی گوش دادن به تقاضاهای جدید برمی‌گردد.[1] (شکل ۶-۹-ب)

هر چند «پروتکل اتصال اولیه» برای سرویس‌دهنده‌هایی که فقط برحسب تقاضا و بصورت موردی اجرا می‌شوند بسیار مفید و کارآمد است ولی گاهی اوقات سرویس‌دهنده‌هایی هستند که مستقل از Process Server کار می‌کنند. به عنوان مثال یک سرویس‌دهندهٔ فایل باید به صورت دائمی بر روی یک ماشین خاص اجرا شده باشد و نمی‌توان آن را برحسب تقاضا و موردی اجرا کرد.

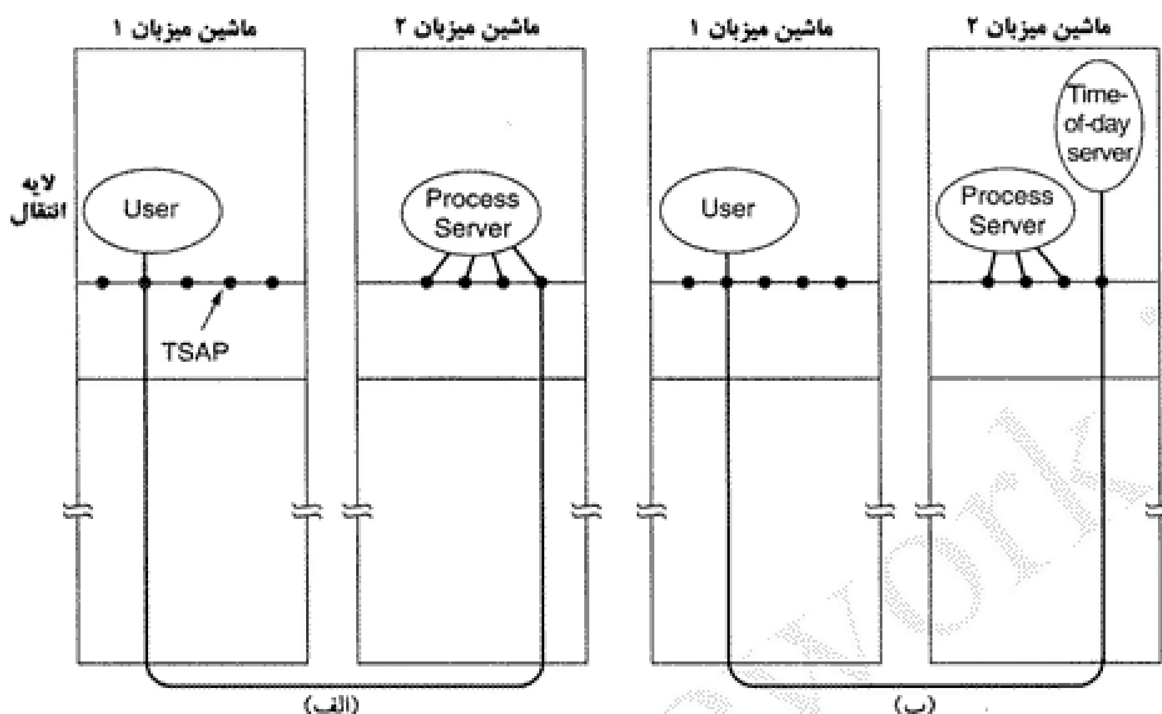
برای حل این مسئله، از الگوی دیگری استفاده می‌شود. در این الگو یک پروسهٔ خاص دیگر به نام Name Server (**سرویس‌دهندهٔ نام**) یا Directory Server (**سرویس‌دهندهٔ دایرکتوری**) بر روی ماشین اجرا می‌شود. برای یافتن آدرس TSAP معادل با نام یک سرویس‌دهنده (مثل سرویس‌دهندهٔ Time-of-Day) ، کاربر ابتدا اتصالی با «سرویس‌دهندهٔ نام» که همیشه به یک شمارهٔ پورت مشهور و شناخته شده گوش می‌دهد، برقرار می‌کند. سپس با ارسال پیامی، نام سرویس‌دهندهٔ مورد نظر خود را سوال کرده و سرویس‌دهندهٔ نام در پاسخ، آدرس TSAP آن را بر می‌گرداند. در اینجا کاربر اتصال خود را با سرویس‌دهندهٔ نام خاتمه داده و اتصال جدیدی با سرویس‌دهندهٔ مورد نظر خود برقرار می‌نماید.[2]

در این مدل هر گاه سرویس‌دهندهٔ جدیدی خلق شود بایستی خودش را در سرویس‌دهندهٔ نام، ثبت کند و نام سرویس (عموماً به صورت یک رشته ASCII) و همچنین TSAP خود را مشخص نماید. سرویس‌دهندهٔ نام این

---

۱. در حقیقت به جای آن که مثلاً n پروسهٔ سرویس‌دهنده به طور همزمان اجرا شود و به پورتهای مورد نظر خود گوش بدهند یک پروسه به نیابت از همهٔ آنها و به طور همزمان به n پورت گوش می‌دهد و در صورت برقراری هر گونه اتصال، پروسهٔ متناظر را اجرا کرده و تماس برقرار شده را به سوی او بر می‌گرداند. -م

۲. این سرویس‌دهندهٔ نام را با سرویس‌دهندهٔ DNS اشتباه نگیرید.



شکل ۶-۹. چگونگی ایجاد اتصال توسط یک پروسهٔ کاربری بر روی ماشین ۱ با سرویس‌دهندهٔ Time of Day (سرویس‌دهندهٔ تاریخ و ساعت) بر روی ماشین ۲.

اطلاعات را در پایگاه اطلاعات داخلی خود درج می‌کند تا در پرس‌وجوهای بعدی پاسخها را بداند.

عملکرد سرویس‌دهندهٔ نام شبیه به عملکرد اپراتورهای راهنما در سیستم تلفن (۱۱۸) است: اسامی را به شماره، ترجمه می‌نماید. اینجا نیز مثل سیستم تلفن دانستن آدرس TSAP سرویس‌دهندهٔ نام الزامی است. این شماره حداقل اطلاعاتی است که هر کسی باید بداند. اگر شما شمارهٔ اپراتور راهنمای تلفن را ندانید نمی‌توانید برای جستجوی شمارهٔ دیگران با این اپراتور تماس بگیرید. اگر فکر می‌کنید شمارهٔ اپراتور راهنما بدیهی است (مثلاً ۱۱۸)، سعی کنید آن را برای یک کشور خارجی نیز امتحان نمائید!

۶-۲-۲ برقراری اتصال (Connection Establishment)

برقراری یک اتصال ساده به نظر می‌رسد ولیکن حقیقتاً بسیار پیچیده است. در نگاه اول شاید اینگونه به نظر برسد که کافی است «واحد انتقال» (Transport Entity) در یک طرف، بستهٔ تقاضای CONNECTION REQUEST TPDU را به سوی مقصد بفرستد و منتظر پاسخ CONNECTION ACCEPTED بماند. اگر شبکه مستعد از بین بردن بسته‌ها، ذخیره و تعلیق آنها یا تولید بسته‌های تکراری (Duplicate) باشد، در برقراری یک اتصال مطمئن، مشکلات جدی رخ می‌دهد و موجب پیچیدگیهای فراوان در پروتکلهای لایهٔ انتقال می‌شود.

زیرشبکه‌ای را مجسم کنید که با ازدحام مواجه شده و پیامهای تصدیق دریافت بسته‌ها (یعنی پیامهای Ack) سر موعد مقرّر بر نمی‌گردند و هر بسته به دلیل اتمام مهلت، دو یا سه بار ارسال می‌شود. در ضمن فرض کنید که زیر شبکه در درون از روش دیتاگرام بهره گرفته و بسته‌ها از مسیرهای متفاوتی به سوی مقصد می‌روند. ممکن است برخی از بسته‌ها در شلوغی ترافیک زیرشبکه گیر افتاده و تحویل آن به مقصد، مدتی طول بکشد. (یعنی در

زیرشبکه ذخیره شده و با تأخیر از آن بیرون بیاید.)

بدترین کابوس ممکن بدین نحو اتفاق می‌افتد: کاربری یک اتصال با سرویس‌دهندهٔ بانک خود برقرار کرده و با ارسال پیامی از آن می‌خواهد که مقدار قابل توجهی پول به حساب یک شخص نه چندان امین! واریز نماید و سپس اتصال را قطع می‌کند. متأسفانه تمام بسته‌ها در این سناریو به صورت تکراری تولید و در زیرشبکه ذخیره می‌شوند.[1] پس از آنکه اتصال قطع شد بسته‌های معلق از زیرشبکه خارج شده و به ترتیب تحویل مقصد می‌شود یعنی به همان ترتیب قبل با بانک یک اتصال برقرار می‌شود، مجدداً تقاضای انتقال پول انجام می‌گیرد و اتصال قطع می‌گردد. بانک، راهی برای تشخیص تکراری بودن این بسته‌ها ندارد و طبعاً باید فرض را بر آن بگذارد که تقاضاها جدید هستند و انتقال پول را انجام بدهد!! تا پایان این بخش به بررسی مسئله بروز بسته‌های تکراری در اثر تأخیر خواهیم پرداخت و بر روی الگوریتمهای برقراری مطمئن اتصال (به نحوی که کابوس فوق هرگز اتفاق نیفتد!) تأکید ویژه خواهیم داشت.

معمای این مسئله در وجود بسته‌هایی تکراری نهفته است که با تأخیر دریافت می‌شوند. به روشهای مختلفی می‌توان به این بسته‌ها حمله کرد و آنها را نابود کرد ولی عملکرد هیچکدام صد در صد مطلوب نیست. یکی از این روشها استفاده از آدرسهای انتقال متغیر با زمان است.[2] در این رویکرد هر گاه به آدرس انتقال [یعنی آدرس TSAP] نیاز شد، یک شمارهٔ جدید تولید می‌شود. وقتی اتصالی قطع شد، آن آدرس کنار گذاشته شده و دیگر از آن استفاده نمی‌شود. این استراتژی موجب می‌شود که مدل مبتنی بر Process Server در شکل ۶-۹، ناممکن و غیرعملی باشد.

رویکرد دیگر آن است که به هر اتصال یک «**شناسهٔ اتصال**» (Connection Identifier) (یا به عبارتی یک شمارهٔ ترتیب که به ازای ایجاد اتصال جدید یک واحد افزایش می‌یابد) به انتخاب تماس گیرنده، منسوب شود. این شناسه در هر TPDU درج و ارسال می‌شود. پس از آن که یک اتصال قطع شد، «واحدهای انتقال» (Transport Entities) در دو طرف، شناسهٔ این اتصال را در جدول شناسه‌های قدیمی درج می‌نمایند. در آینده وقتی تقاضای جدیدی مبنی بر برقراری یک اتصال جدید از راه می‌رسد، ابتدا باید به کمک این جدول بررسی شود که مبادا متعلق به اتصالی باشد که قبلاً قطع شده است.

متأسفانه این روش نیز یک اشکال اساسی دارد: هر یک از «واحدهای انتقال» موظفند پیشینهٔ این شناسه‌ها را برای همیشه در جدول خود نگاه دارند. اگر ماشینی از کار بیفتد و حافظهٔ خود را از دست بدهد دیگر نمی‌تواند تشخیص بدهد که کدامیک شناسه‌های اتصال تکراری هستند.

به جای اینها، باید از راه دیگری وارد شویم: به جای آن که اجازه بدهیم بسته‌ها در زیرشبکه عمر جاودان داشته باشند بایستی مکانیزمی اتخاذ کنیم که بسته‌های با عمر زیاد و سرگردان نابود شوند. اگر مطمئن شویم که بسته‌ها هیچگاه بیش از یک مدت معین عمر نمی‌کنند، مشکل فوق‌الذکر قابل کنترل خواهد شد.

طول عمر بسته‌ها را می‌توان با تکنیکهای زیر در حد مشخصی محدود کرد:

۱. طراحی محدود شدهٔ زیرشبکه (Restricted Subnet Design)
۲. درج شمارندهٔ گام در هر بسته (Hop Counter)
۳. درج مهر زمان در هر بسته (Timestamp)

---

۱. چراکه مثلاً هر بسته پس از ارسال در زیرشبکه بیش از حدّ معطل شده و مهلت فرستنده به سرآمده و یک نسخهٔ دیگر از آنرا فرستاده است؛ غافل از آنکه هر دو بستهٔ تکراری نهایتاً با تاخیر به مقصد خواهند رسید. -م

۲. بعبارت دیگر آدرس TSAP برنامهٔ مشتری در هر بار ایجاد اتصال عوض شود. -م

اولین روش، متضمن استفاده از هر راهکاری است که از چرخیدن بسته‌ها در یک حلقه جلوگیری کرده و همچنین تأخیر ناشی از ازدحام را بر روی طولانی‌ترین مسیر ، در حد معینی محدود نگه دارد. دومین روش، مستلزم آن است که در هر بسته یک شمارندهٔ گام قرار داده شده و در مبداء یک مقدار اولیه مناسب به آن منتسب گردد و به ازای عبور از هر مسیریاب یک واحد از آن کم شود. پروتکل لایهٔ شبکه بسته‌هایی را که مقدار این شمارنده در آنها به صفر رسیده حذف می‌نماید. سومین روش نیازمند آن است که هر بسته ، زمان ایجاد خود را با خود حمل نماید و بر این اساس مسیریابها طبق توافق، بسته‌هایی که بیش از یک مدت معین قدیمی شده‌اند را حذف کنند. استفاده از این روش مستلزم آن است که ساعت تمام مسیریابها به دقت با هم تنظیم شده باشد ولیکن این خودش کار ساده‌ای نیست مگر آن که عمل تنظیم ساعتها از بیرون شبکه انجام شود. (مثلاً به کمک سیستم ماهواره‌ای GPS یا ایستگاههای رادیویی خاص تا ساعت دقیق را به همه اعلام کند.)

در عمل نه تنها لازم است که تضمین کنیم بسته‌های قدیمی نابود شده‌اند بلکه باید تمام بسته‌های اعلام وصول آنها (Ack's) نیز از بین رفته باشد. با این استدلال، زمانی بنام T را معرفی می‌کنیم که مقدار آن چند برابر حداکثر طول عمر واقعی بسته‌هاست. (چند برابر بودن، به پروتکل مورد استفاده بستگی دارد.) اگر پس از ارسال یک بسته به اندازهٔ زمان T صبر کنیم، می‌توانیم مطمئن شویم که نه تنها تمام آثار آن بسته از زیرشبکه پاک شده بلکه حتی پیامهای Ack آن نیز از بین رفته است.

با محدود شدن طول عمر بسته‌ها، می‌توان روشی مطمئن و بی‌خطا برای برقراری اتصال ابداع کرد. روشی را که در ادامه معرفی می‌نمائیم، پیشنهاد تاملینسون (Tomlinson/۱۹۷۵) است. این روش مشکل مذکور را حل می‌کند ولی پیچیدگیهای خاص خود را دارد. این روش بعداً توسط دو نفر به نامهای Sunshine و Dalal (۱۹۷۸) بهینه شد. گونه‌های متفاوتی از این روش در عمل مورد استفاده قرار گرفته است (مثلاً در TCP).

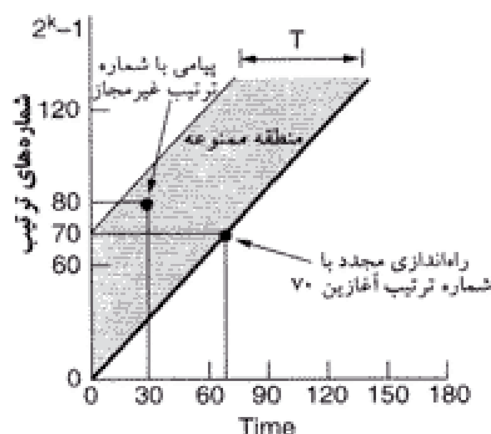
برای آن که مشکل از کار افتادن ماشین و از دست رفتن محتوای حافظه رفع شود، تاملینسون پیشنهاد کرد که در هر ماشین یک ساعت تعبیه شود. لازم نیست که ساعت ماشینهای مختلف با یکدیگر تنظیم شده باشند. فرض بر آن است که این ساعت همانند یک شمارندهٔ باینری عمل می‌کند که در فواصل زمانی مشخص یک واحد افزایش می‌یابد. تعداد بیتهای شمارندهٔ ساعت باید بزرگتر یا مساوی با تعداد بیتهای شمارهٔ ترتیب بسته‌ها باشد. مورد آخر و مهمتر از همه آنکه ساعت مذکور باید حتی وقتی ماشین از کار می‌افتد یا خاموش می‌شود، کار کند.

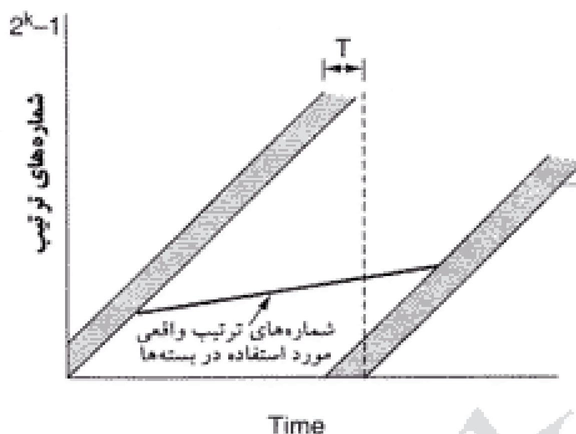
ایدهٔ اصلی این روش آن است که مطمئن شویم در یک زمان، دو TPDU با شمارهٔ مشابه تولید و ارسال نمی‌شود. برای ایجاد یک اتصال، از k بیت کم ارزش ساعت به عنوان شمارهٔ ترتیب اولیه استفاده می‌شود. (فیلد شمارهٔ ترتیب بسته‌ها نیز k بیتی است.) بنابراین برخلاف پروتکلهای معرفی شده در فصل ۳، در هر اتصال، TPDUها از شمارهٔ ترتیب متفاوتی استفاده می‌کنند. فضای در نظر گرفته شده برای شمارهٔ ترتیب باید آنقدر بزرگ باشد که در صورت برگشت مقدار این شماره به صفر[۱] ، TPDUهای قدیمی با همین شماره‌ها، مدتها پیش از بین رفته باشند. رابطهٔ خطی بین «زمان» و «شمارهٔ ترتیب اولیه» در شکل ۶-۱۰ نشان داده شده است.

به محض آن که «واحدهای انتقال» (Transport Entity) در دو طرف، بر روی شمارهٔ ترتیب اولیه به توافق رسیدند، می‌توان از هر پروتکلی نظیر «پروتکل پنجرهٔ لغزان» (Sliding Window Protocol) برای کنترل جریان استفاده کرد. در دنیای واقعی، منحنی شمارهٔ ترتیب (که در شکل با خطوط پر رنگ نشان داده شده) خطی نیست بلکه حالت پلکانی دارد زیرا شمارش ساعت به صورت گسسته (مثلاً در هر هزارم ثانیه) صورت می‌گیرد. برای سادگی از این جزئیات صرفنظر خواهیم کرد.

---

۱. به برگشت یک شمارنده به صفر پدیده Wrapping Around گفته می‌شود. -م





شکل ۶-۱۰. (الف) شمارهٔ ترتیب TPDUها نباید به منطقهٔ ممنوعه وارد شود. (ب) مشکل سنکرون‌سازی مجدّد شماره‌ها

مشکل زمانی بروز می‌کند که ماشین میزبان از کار بیفتد: وقتی از نو شروع به کار می‌کند، «واحد انتقال» از شمارهٔ ترتیب قبلی خود مطلع نیست. یک راه حل آن است که «واحد انتقال» پس از راه‌اندازی مجدد مدت T ثانیه بیکار بماند تا تمام TPDUهای قدیمی از بین بروند. ولی در شبکه‌های بزرگ و پیچیده، مقدار T می‌تواند بسیار بزرگ باشد و بدین ترتیب استراتژی فوق چندان جالب و مفید نیست.

برای آن که نیازی به این زمان مرده نباشد باید محدودیت جدیدی بر روی شماره‌های ترتیب گذاشته شود. با یک مثال به توضیح محدودیت جدید می‌پردازیم: فرض کنید که T (یعنی حداکثر طول عمر بسته‌ها) ۶۰ ثانیه باشد و ساعت سیستم در هر ثانیه یک تیک بزند. به نحوی که در شکل ۶-۱۰-الف با خط تیره رنگ نشان داده شده، شماره ترتیب هر اتصال که در زمان x آغاز می‌شود، همان x است. فرض کنید که در لحظهٔ t=30 یک بستهٔ TPDU معمولی با شمارهٔ ترتیب ۸۰ بر روی اتصال ۵ (که قبلاً ایجاد شده) ارسال گردد. این بسته را TPDU X بنامید. بلافاصله پس از ارسال TPDU X، ماشین میزبان از کار افتاده و سریعاً راه‌اندازی می‌شود. در لحظهٔ t=60 این ماشین اتصالات ۰ تا ۴ را از نو ایجاد می‌کند. در لحظهٔ t=70، اتصال ۵ را نیز از نو برقرار می‌سازد و طبعاً شمارهٔ ترتیب بسته‌ها از ۷۰ شروع می‌شود. در ۱۵ ثانیهٔ بعدی، این ماشین TPDUهای ۷۰ تا ۸۰ را ارسال می‌نماید. در لحظهٔ t=85 یک TPDU جدید با شمارهٔ ۸۰ بر روی اتصال ۵ ارسال و درون زیرشبکه تزریق می‌شود.[1] متأسفانه TPDU X هنوز در زیرشبکه حضور دارد و اگر قبل از TPDU شمارهٔ ۸۰ برسد، پذیرفته شده و بستهٔ جدید (یعنی TPDU 80) به عنوان بستهٔ تکراری حذف خواهد شد.

برای پیشگیری از چنین مشکلاتی، باید راهکاری اتخاذ شود تا شماره‌های ترتیبی که در TPDUهای جدید درج می‌شود تا قبل از انقضای زمان T بهیچوجه مشابه با شماره‌های قبلی نباشد. در شکل ۶-۱۰-الف شماره‌های ترتیبی که در هر لحظه از زمان استفاده از آنها مجاز نیست، تحت عنوان «**ناحیهٔ ممنوعه**» (Forbidden Region) نشان داده شده است.[2] قبل از ارسال هر گونه TPDU بر روی یک اتصال، «واحد انتقال» باید مقدار باینری ساعت

---

۱. فقط شمارهٔ آغازین از ساعت سیستم اخذ می‌شود و برای بسته‌های بعدی این شماره فقط افزایش می‌یابد. -م

۲. به خاطر داشته باشید که ملاک انتخاب شمارهٔ ترتیب برای بسته‌های TPDU در ماشینی که از کار افتاده و سریعاً راه‌اندازی شده ساعت سیستم است. به همین دلیل نمودار «ناحیهٔ ممنوعه» در دو محور زمان و شمارهٔ ترتیب ترسیم شده است. تعبیر ساده‌تر ناحیهٔ ممنوعه آن است که هر ماشین موظف است برای شماره‌گذاری بسته‌ها، از عددی شروع کند که نسبت به مقدار

فعلی را خوانده و بررسی کند که مبادا در ناحیهٔ ممنوعه قرار گرفته باشد.

پروتکل فوق از دو جهت به دردسر می‌افتد: اگر ماشین میزبان تعداد بسیار زیادی بسته را با سرعت فوق‌العاده بالا بر روی اتصال ایجاد شده، بفرستد، آنگاه منحنی «شمارهٔ ترتیب برحسب زمان» شیب بیشتری نسبت به منحنی «شمارهٔ ترتیب اولیه» در شکل ۶-۱۰ پیدا خواهد کرد.[1] این مسئله بدین معناست که حداکثر نرخ ارسال بسته بر روی یک اتصال، نباید از یک TPDU در هر تیک ساعت تجاوز کند. همچنین «واحد انتقال» در ماشینی که از کار افتاده و مجدداً راه‌اندازی شده بایستی قبل از باز کردن هر اتصال جدید آنقدر صبر کند تا ساعت سیستم، تیک بزند مبادا شمارهٔ انتخابی، تکراری شود. هر دوی این موارد فوق، ایجاب می‌کند که تیکهای ساعت بسیار کوتاه باشد. (مثلاً هر چند میکروثانیه یکبار یا حتی کمتر تیک بزند.)

متأسفانه، ورود به ناحیهٔ ممنوعه (در اثر سرعت بسیار زیاد فرستنده) تنها دلیل بروز مشکل نیست. در شکل ۶-۱۰-ب می‌بینیم که اگر نرخ ارسال کمتر از نرخ تیک زدن ساعت باشد، ورود به ناحیهٔ ممنوعه از سمت چپ اتفاق می‌افتد.[2] هر چه شیب منحنی شمارهٔ ترتیب برحسب زمان، بیشتر باشد بروز این مشکل به تعویق خواهد افتاد. همانگونه که قبلاً اشاره کردیم، قبل از ارسال هر TPDU، «واحد انتقال» باید بررسی کند که مبادا به ناحیهٔ ممنوعه وارد شود و اگر اینچنین است بایستی ارسال TPDU را به اندازهٔ T ثانیه به تعویق بیندازد یا آن که شماره‌های ترتیب را از نو سنکرون نماید.

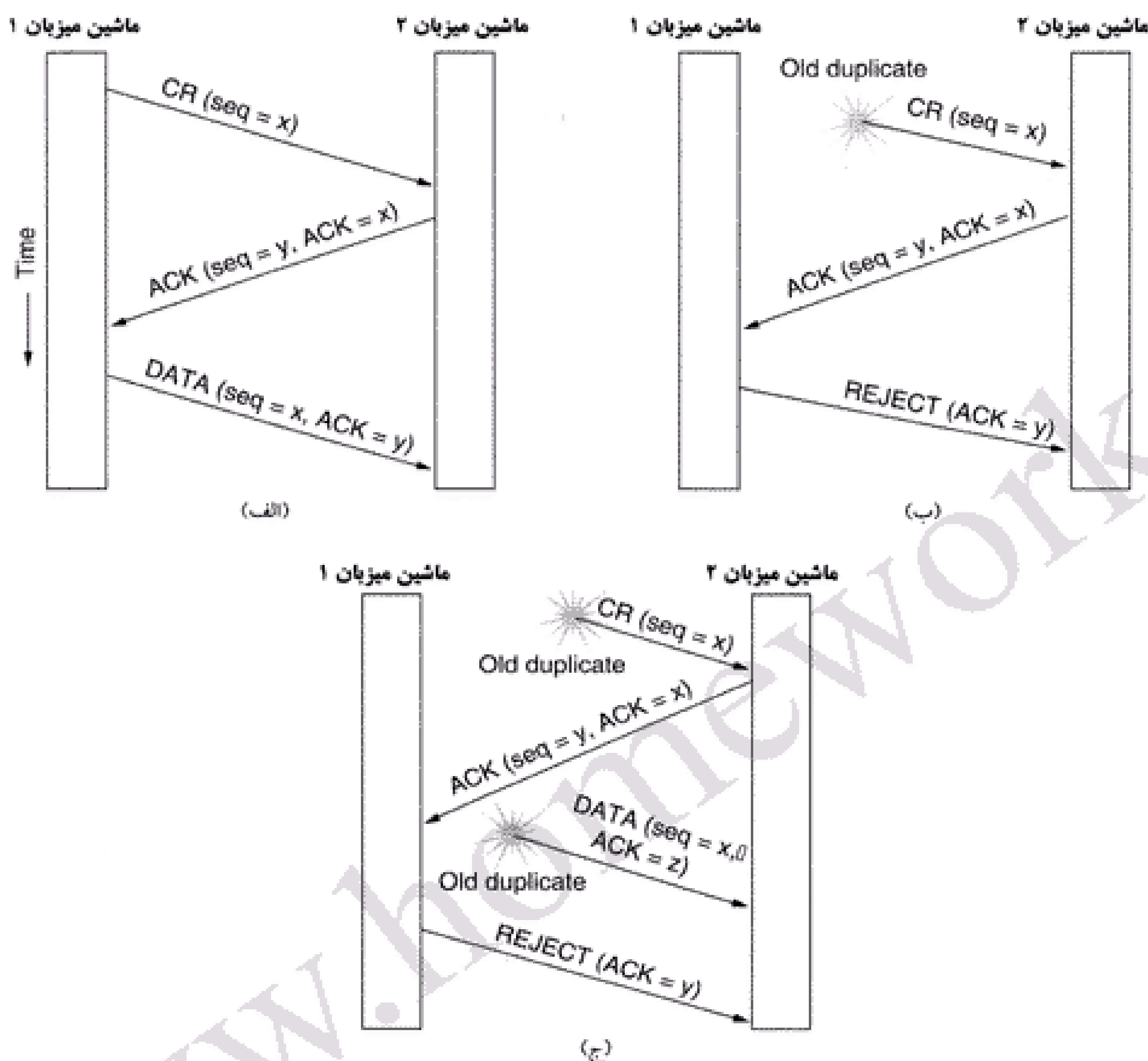
روش مبتنی بر ساعت اگرچه مشکل بسته‌های تکراری (ناشی از تأخیر) را حل می‌کند ولیکن برای آن که این راهکار مفید فایده باشد، ابتدا باید «اتصال» مورد نظر ایجاد شود. از آنجایی که بسته‌های کنترلی TPDU نیز ممکن است با تأخیر مواجه شوند، این استعداد وجود دارد که طرفین، در حین توافق بر روی شمارهٔ ترتیب با مشکل مواجه شوند. به عنوان مثال فرض کنید که با ارسال یک بستهٔ CONNECTION REQUEST TPDU از طرف ماشین ۱ به ماشین ۲ (به همراه شمارهٔ پورت مقصد و شمارهٔ ترتیب اولیه) اقدام به برقراری یک اتصال شود. گیرنده یعنی ماشین میزبان ۲، نیز با ارسال بستهٔ کنترلی CONNECTION ACCEPTED TPDU، دریافت بستهٔ قبلی را تصدیق (Ack) می‌نماید. اگر بستهٔ CONNECTION REQUEST TPDU از دست رفته و نسخهٔ تکراری آن با تأخیر به ماشین ۲ برسد این اتصال به نحو نادرستی برقرار خواهد شد.

برای حل این مشکل، تاملینسون (۱۹۷۵) روشی به نام **«دست تکانی سه مرحله‌ای»** (Three Way Handshake) پیشنهاد کرد. برای برقراری اتصال، در این پروتکل نیازی نیست که طرفین بر روی شمارهٔ ترتیب مشترک و مشابهی توافق نمایند، فلذا در این پروتکل می‌توان از روشی به غیر از روش مبتنی بر ساعت سراسری سیستم، استفاده کرد. روال معمولی برقراری یک اتصال (که در آن ماشین ۱ شروع کنندهٔ آن است)، در شکل ۶-۱۱-الف نشان داده شده است: ماشین ۱، یک شمارهٔ ترتیب دلخواه مثل x انتخاب کرده و با ارسال بستهٔ CONNECTION REQUEST TPDU آن را به طرف مقابل اعلام می‌کند. ماشین ۲ نیز با ارسال بسته‌ای به نام ACK TPDU، ضمن تصدیق شمارهٔ ترتیب x شمارهٔ ترتیب بسته‌های خودش (یعنی y) را به همتای خود اعلام می‌دارد. نهایتاً ماشین ۱، شمارهٔ ترتیب انتخاب شده توسط ماشین ۲ را (در اولین بستهٔ داده‌ای که ارسال می‌کند)، تصدیق خواهد نمود.

عددی ساعت فعلی، T واحد جلوتر باشد. -م

۱. به عبارت بهتر به دلیل آن که شماره‌های ترتیب در هر تیک ساعت یکبار افزایش می‌یابند لذا اگر بسته‌های TPDU با سرعت بیشتری نسبت به تیکهای ساعت تولید شوند اجباراً برخی از شماره‌های ترتیب تکراری می‌شوند. -م

۲. به عبارت بهتر اگر نرخ ارسال کمتر از نرخ تیک زدن باشد، چون شماره‌های ترتیب باینری و با طول محدود است، به دلیل بالا بودن نرخ تیک زدن، این شماره به صفر برگشته و از نو وارد ناحیهٔ ممنوعه می‌شود. -م



شکل ۶-۱۱. سه سناریوی مختلف برای برقراری اتصال طبق روش «دست تکانی سه‌مرحله‌ای». CR مخفف CONNECTION REQUEST است. (الف) عملکرد طبیعی. (ب) نسخهٔ تکراری و قدیمی بستهٔ CONNECTION REQUEST به ناگاه ظاهر می‌شود. (ج) نسخهٔ تکراری بسته‌های CONNECTION REQUEST و همچنین ACK دریافت می‌شوند.

حال اجازه بدهید بررسی کنیم که این پروتکل در برخورد با نسخه‌های تکراری بسته‌های کنترلی TPDU (که در اثر تأخیر زیرشبکه تولید می‌شوند) چگونه کار می‌کند. در شکل ۶-۱۱-ب، فرض شده که بستهٔ CONNECTION REQUEST که نسخه‌ای تکراری و بجا مانده از یک اتصال قدیمی است، با تأخیر دریافت می‌شود. این TPDU (بدون آن که ماشین ۱ از آن اطلاع داشته باشد) به ماشین ۲ می‌رسد و طبعاً ماشین ۲ با ارسال بستهٔ کنترلی ACK TPDU به آن پاسخ می‌دهد. در حقیقت ماشین ۲ با ارسال این بسته خواسته که بررسی کند آیا ماشین ۱ واقعاً تقاضای برقراری اتصالی جدید داده است. وقتی ماشین ۱، تلاش ماشین ۲ برای برقراری این اتصال را رد می‌کند، ماشین ۲ می‌فهمد که با دریافت یک نسخهٔ تکراری از بستهٔ تقاضا، فریب خورده و از پذیرش اتصال امتناع می‌کند. بدین ترتیب، بسته‌های تکراری (ناشی از تأخیر) خطری ندارند.

بدترین حالت زمانی است که هر دو نسخهٔ تأخیر یافته و تکراری CONNECTION REQUEST و ACK

در زیرشبکه شناور باشند. این حالت در شکل ۶-۱۱-ج دیده می‌شود. همانند مثال قبلی، ماشین ۲ یک نسخهٔ تکراری از بستهٔ کنترلی CONNECTION REQUEST دریافت کرده و بدان پاسخ می‌دهد. در اینجا درک این موضوع اهمیت حیاتی دارد که ماشین ۲ با ارسال y پیشنهاد کرده شمارهٔ ترتیب ترافیک خودش (ماشین ۲) به ماشین ۱ از شماره y شروع شود و y را به گونه‌ای انتخاب کرده که هیچ بستهٔ TPDU یا بستهٔ Ack با شمارهٔ y در شبکه سرگردان نمانده باشد.[1] وقتی بستهٔ تأخیر یافتهٔ دوم (یعنی بسته Ack تکراری و باقیمانده از قبل) دریافت می‌شود، با توجه بدان که به جای y، شمارهٔ ترتیب z مورد تأئید قرار گرفته، ماشین ۲ می‌فهمد که این بسته قدیمی است. نکتهٔ مهم در اینجاست که **هیچ ترکیبی از بسته‌های قدیمی نمی‌تواند باعث شکست پروتکل و ایجاد اتصال ناخواسته گردد.**[2]

۶-۲-۳ خاتمهٔ اتصال

خاتمه دادن به یک اتصال ساده‌تر از ایجاد آن است ولیکن کار به آن سادگی هم که انتظار می‌رود، نیست. قبلاً اشاره کردیم که برای خاتمه دادن به یک اتصال دو سبک وجود دارد: متقارن (Symmetric) و نامتقارن (Asymmetric). روش نامتقارن همان روشی است که در سیستم تلفن هم وجود دارد: وقتی یکی از طرفین گوشی را قطع می‌کند، تماس (اتصال) قطع می‌شود. در خاتمهٔ متقارن، با هر اتصال به صورت دو «ارتباط یکطرفه» رفتار می‌شود که هر یک از آنها به صورت مستقل و مجزا خاتمه می‌یابند.

روش خاتمهٔ نامتقارن، فرآیندی ناگهانی است و می‌تواند منجر به از دست رفتن بخشی از داده‌ها شود. به سناریوی شکل ۶-۱۲ دقت کنید: پس از برقراری اتصال، ماشین ۱ یک بستهٔ TPDU برای ماشین ۲ می‌فرستد و به درستی تحویل می‌شود. سپس ماشین ۱ بسته‌ای دیگر ارسال می‌کند. متأسفانه قبل از تحویل بستهٔ دوم، ماشین ۲ تقاضای قطع ارتباط (DISCONNECT) را صادر می‌نماید. این کار موجب می‌شود که اتصال قطع شود و بستهٔ دوم از دست برود.

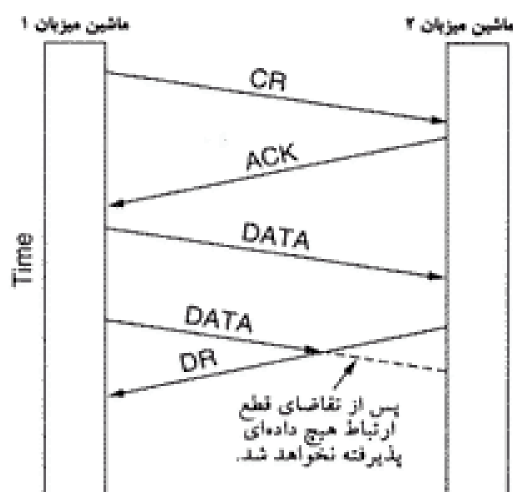
بدیهی است برای آنکه هیچ داده‌ای از دست نرود به پروتکل پیچیده‌تری برای خاتمه دادن به اتصال نیاز است. یک راهکار آن است که از روش متقارن برای قطع اتصال استفاده شود تا اتصال در هر یک از دو طرف به صورت مستقل از دیگری خاتمه یابد. در این روش ماشینی که با ارسال بستهٔ کنترلی DISCONNECT TPDU تقاضای ختم ارتباط می‌کند کماکان به دریافت داده ادامه خواهد داد.

روش خاتمهٔ متقارن (Symmetric Release) زمانی خوب کار می‌کند که هر پروسه حجم ثابت و مشخصی داده برای ارسال داشته باشد و دقیقاً بداند چه زمانی آنها را ارسال کرده است. در شرایطی بغیر از این، تعیین آن که آیا کار به اتمام رسیده و اتصال بایستی قطع شود، ساده نخواهد بود. شاید یک پروتکل پیشنهادی آن باشد که ماشین ۱ بسادگی بگوید: «کار من تمام است. کار شما چطور؟» هر گاه ماشین ۲ پاسخ بدهد که «کار من نیز تمام شد؛ خداحافظ!»، اتصال بین آنها بطور مطمئن خاتمه یابد.

---

۱. یعنی شمارهٔ ترتیب بسته‌های خودش را با اطمینان از عدم وجود بسته‌هایی تکراری و سرگردان با همین شماره‌ها، انتخاب می‌کند. -م

۲. فرآیند فوق را می‌توان در یک عبارت ساده‌تر، بدینگونه خلاصه نمود: ماشین ۱ با ارسال بستهٔ REQUEST اعلام می‌دارد که تمایل به برقراری یک اتصال دارد و در ضمن مایل است بسته‌های خود را از شمارهٔ x شروع کند. x شماره‌ای تصادفی است و از تکراری نبودن آن اطمینان دارد چرا که مثلاً در دو دقیقهٔ قبل تاکنون از آن استفاده نکرده است. ماشین ۲ در پاسخ، ضمن تأئید شمارهٔ x، اعلام می‌دارد که با برقراری اتصال موافق است و او نیز بسته‌هایش را از شمارهٔ y آغاز می‌کند. y نیز شماره‌ای تصادفی و غیرتکراری انتخاب می‌شود. در مرحلهٔ سوم x و y به تأئید نهایی می‌رسد. بدین ترتیب، از آنجایی که بسته‌های تکراری، شماره‌های x یا y آنها فرق می‌کند ـ فارغ از آن که از نوع REQUEST باشند یا Ack ـ پذیرفته نخواهند شد. -م

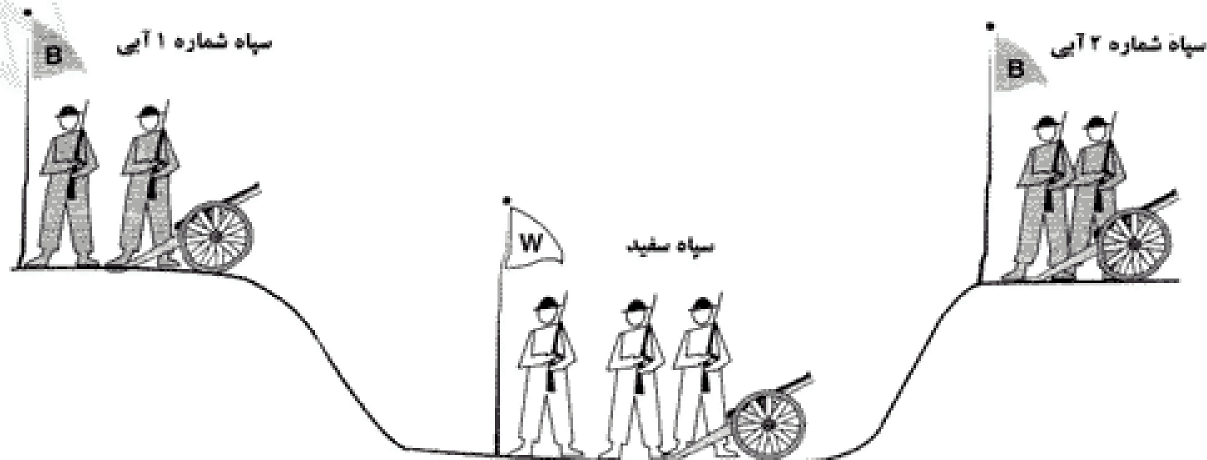


شکل ۶-۱۲. قطع ناگهانی اتصال و از دست رفتن بخشی از داده‌ها.

متأسفانه این پروتکل همیشه کار نخواهد کرد؛ در این خصوص مسئله مشهوری وجود دارد که مشکل پروتکل را مشخص می‌نماید و به نام «**مسئله دو سپاه**» (Two Army Problem) معروف است. تجسم کنید که سپاه سفید در پایین یک درّه اردو زده است. به گونه‌ای که در شکل ۶-۱۳ نشان داده شده، دو سپاه آبی، دره را از دو طرف محاصره کرده‌اند. نفرات سپاه سفید از یک سپاه آبی بیشتر است ولی مجموع نفرات دو سپاه آبی از سپاه سفید افزونتر می‌شود. اگر هر یک از سپاهیان آبی به تنهایی حمله را آغاز کنند شکست خواهند خورد مگر آن که هر دو سپاه آبی در یک زمان یورش ببرند.

سپاهیان آبی می‌خواهند که حملهٔ خود را با یکدیگر هماهنگ نمایند ولیکن تنها راه ارتباط آنها، فرستادن یک پیک به پایین و عبور از درّه است. کاری که ممکن است منجر به دستگیری پیک و از دست رفتن پیغام شود. (به عبارتی آنها با یک کانال ارتباطی نامطمئن مواجه هستند.) سؤال این است که آیا پروتکلی وجود دارد که براساس آن سپاهیان آبی پیروز شوند؟

فرض کنید که فرمانده سپاه یکم آبی پیامی بدین مضمون ارسال می‌کند: «من پیشنهاد می‌کنم که حمله را در سپیده دم روز ۲۹ مارچ شروع کنیم. نظر شما چیست؟» فرضاً این پیام به سلامت می‌رسد و فرمانده سپاه دوم آبی



شکل ۶-۱۳. مسئلهٔ دو سپاه.

موافقت کرده و پاسخ او نیز به سپاه یکم آبی بر می‌گردد. آیا حمله در موعد مقرر انجام می‌شود؟ شاید نه! چرا که فرماندهٔ سپاه دوم نمی‌داند که آیا پاسخ او به سلامت رسیده است یا خیر! به گمان او اگر در راه بازگشت بلایی بر سر پیک آمده باشد، سپاه یکم آبی حمله نمی‌کند، بنابراین شروع چنین نبردی احمقانه خواهد بود!!

حال بیائید پروتکل فوق را از دو مرحله به سه مرحلهٔ دست تکانی (Three Way Handshake) بهبود ببخشیم. فرماندهی که پیشنهاد اول را می‌دهد، باید در نهایت پاسخ طرف مقابل خود را تصدیق کند. حال فرض کنید که هیچ پیامی از دست نرود و سپاه دوم آبی، پیام تصدیق را دریافت نماید ولیکن در اینجا فرماندهٔ سپاه یکم آبی به تردید می‌افتد! چرا که مطمئن نیست که آیا پیام تصدیق او به سپاه دوم رسیده است یا خیر. اگر نرسیده باشد سپاه دوم آبی حمله نخواهد کرد! شاید بتوان پروتکل را چهار مرحله‌ای کرد ولیکن باز هم کمک چندانی نمی‌کند.[1] در حقیقت می‌توان ثابت کرد که هیچ پروتکلی که بتواند بدون تردید عمل کند وجود ندارد. فرض کنید که چنین پروتکلی وجود داشته باشد. در اینجا یا آخرین پیام پروتکل حیاتی هست یا نیست: اگر حیاتی نیست باید آن را حذف کرد. در حقیقت تمام پیامهای غیرضروری باید حذف شوند تا در پروتکل فقط پیامهای ضروری باقی بمانند. حالا چه اتفاقی می‌افتد اگر آخرین پیام، به مقصد نرسد؟ باید گفت که این پیام ضروری بوده و اگر از دست برود حمله شروع نخواهد شد. از آنجایی که فرستندهٔ آخرین پیام از سرنوشت پیامش مطمئن نیست، خود را با شروع حمله به خطر نخواهد انداخت. بدتر آن که، سپاه طرف مقابل نیز در همین گمان که ممکن است طرف مقابل او حمله نکند، عملیات را آغاز نخواهد کرد!!

برای آن که بتوانید مسئلهٔ دو سپاه را به مسئلهٔ خاتمهٔ اتصال ربط بدهید و با هم قیاس کنید به جای واژهٔ «حمله»، واژهٔ «قطع اتصال» بگذارید. اگر هیچیک از طرفین متقاعد نشوند که طرف مقابل واقعاً آمادهٔ قطع اتصال است، اتصال هیچگاه خاتمه نمی‌یابد!

در عمل، خطر خاتمهٔ یک اتصال از خطر حمله به سپاه سفید کمتر است!! و در ضمن یکی از طرفین آمادگی بیشتری جهت پذیرش چنین خطری را دارد، لذا وضعیت فوق چندان هم ناامیدکننده نیست! در شکل ۶-۱۴ چهار سناریوی مختلفی که ممکن است در پروتکل قطع سه مرحله‌ای اتصال، اتفاق بیفتد به تصویر کشیده شده است. اگرچه این پروتکل چندان مصون از خطا نیست ولیکن معمولاً کفایت می‌کند.

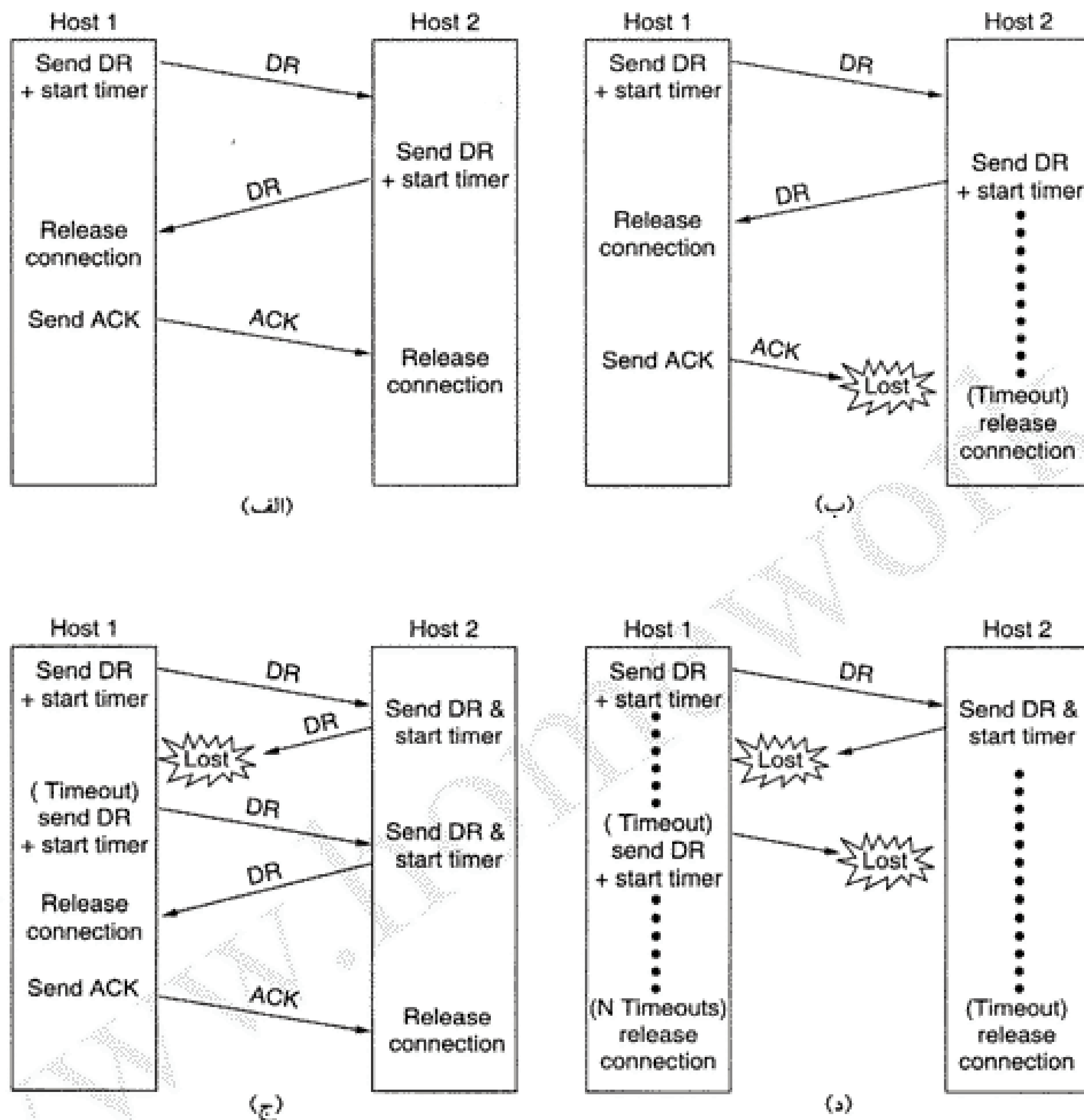
در شکل ۶-۱۴-الف، حالت طبیعی قطع اتصال را مشاهده می‌کنیم که در آن ماشین ۱ با ارسال بستهٔ کنترلی DR TPDU (DISCONNECT REQUEST) تقاضای خاتمهٔ اتصال را به طرف مقابل خود ارسال کرده است. وقتی در طرف مقابل این بسته دریافت شود، گیرندهٔ آن با ارسال متقابل بستهٔ DR TPDU ، پاسخ داده و یک تایمر را فعال می‌کند تا در صورت از بین رفتن این بسته از تایمر کمک بگیرد. وقتی این بستهٔ DR دریافت شود، فرستندهٔ اصلی با بازگرداندن بستهٔ ACK TPDU ، به این فرآیند پایان داده و اتصال از طرف او قطع می‌شود. در نهایت وقتی بستهٔ ACK TPDU به طرف مقابل می‌رسد او نیز به این اتصال خاتمه می‌دهد. «**خاتمهٔ یک اتصال**» بدین معناست که «واحد انتقال» (Transport Entity) تمام اطلاعاتی را که در خصوص آن اتصال در «جدول اتصالات باز» ذخیره نموده پاک کند و به نحوی به صاحب آن اتصال (کاربر) اطلاع بدهد. این عمل با صدور تابع DISCONNECT که توسط کاربر لایهٔ انتقال انجام می‌گیرد متفاوت است.

شکل ۶-۱۴-ب بیانگر حالتی است که بستهٔ ACK TPDU از دست رفته است (این موضوع توسط تایمر آشکار می‌شود). وقتی مهلت تایمر مربوطه منقضی گردد، خواه ناخواه اتصال مربوطه قطع می‌شود.

اکنون حالتی را در نظر بگیرید که در آن دومین بستهٔ کنترلی DR از بین می‌رود. کاربری که در ابتدا اقدام به تقاضای قطع اتصال کرده پاسخ مورد نظر خود را دریافت نخواهد کرد و این کار را از نو تکرار می‌نماید. شکل

---

۱. زیرا همیشه در آخرین پیام شک و تردید وجود دارد و یک دور باطل ایجاد می‌شود. -م



شکل ۶-۱۴. چهار سناریوی مختلف خاتمه دادن به اتصال. (الف) حالت طبیعی دست‌تکانی سه‌مرحله‌ای. (ب) آخرین ACK از دست رفته است. (ج) پاسخ از دست رفته است. (د) اولین پاسخ و تمام تقاضاهای بعدی قطع ارتباط (یعنی تمام DRها) از دست رفته‌اند.

۶-۱۴-ج چنین حالتی را به تصویر کشیده است (با این فرض که بعد از بستهٔ DR دوم، هیچ بستهٔ دیگری از دست نرود و تمام بسته‌های TPDU به سلامت و سر وقت تحویل شود.)

در آخرین سناریو که در شکل ۶-۱۴-د نشان داده شده، همهٔ شرایط مثل شکل ۶-۱۴-ج است با این تفاوت که فرض کرده‌ایم به غیر از بستهٔ کنترلی DR اول، تمام بسته‌های بعدی نیز از بین رفته‌اند و هر گونه تلاش جهت ارسال مجدد با شکست مواجه شده است. پس از N بار تلاش پیاپی، فرستنده ناامید شده و به ناچار اتصال را قطع می‌نماید. عاقبت، تایمر گیرندهٔ طرف مقابل نیز منقضی شده و او نیز اتصال را خاتمه می‌دهد.

اگرچه برای دنیای عمل، این پروتکل کفایت می‌کند ولی از دیدگاه تئوری، اگر بستهٔ DR اول و تمام N تکرار

بعدی آن از بین بروند، پروتکل با شکست مواجه می‌شود، زیرا اگرچه فرستندهٔ اولین بسته ناامید شده و پس از انقضای مهلت مقرر، به اتصال خاتمه می‌دهد ولیکن طرف مقابل او از تمام این تلاشها و ارسال پیاپی بسته‌ها بی‌خبر است و طبعاً فعال باقی می‌ماند. در چنین شرایطی، اتصال به صورت «**نیمه باز**» (Half-Open) باقی می‌ماند.

برای اجتناب از چنین وضعیتی می‌توان فرستنده را وادار کرد که حتی پس از N تلاش مجدد باز هم ناامید نشود و آنقدر کار را ادامه بدهد تا بالاخره پاسخی دریافت نماید. ولیکن اگر مهلت طرف مقابل منقضی شده و به اتصال خاتمه بدهد، فرستنده هرگز پاسخی دریافت نخواهد کرد و طبعاً در حلقهٔ بی‌نهایت می‌افتد. اگر هم به طرف مقابل اجازه ندهیم که در اثر انقضای مهلت اتصال را قطع کند آنگاه در شرایطی که در شکل ۶-۱۴-د نشان داده شده، پروتکل قفل خواهد شد.

یکی از روشهای حذف اتصالات نیمه باز وضع این قانون است که اگر پس از گذشت زمان معینی (برحسب ثانیه) هیچ بستهٔ TPDU دریافت نشد، اتصال باید به صورت خودکار قطع شود. بدین ترتیب اگر یکی از طرفین به صورت یکجانبه اتصال را قطع نماید، طرف مقابل متوجه عدم فعالیت او شده و او نیز به اتصال خاتمه می‌دهد. البته اگر بخواهیم چنین قانونی اعمال شود لازم است که «واحد انتقال» دارای تایمر خاصی باشد که با ارسال یک TPDU شروع به اندازه‌گیری زمان می‌نماید. اگر پس از ارسال آن TPDU، مهلت تایمر منقضی شد و بستهٔ دیگری جهت ارسال وجود نداشت فرستنده موظف به ارسال یک بستهٔ پوچ (بدون داده) برای طرف مقابل است تا از قطع شدن اتصال (در اثر انقضای مهلت مقرر) جلوگیری نماید. در سمت مقابل نیز اگر قانون قطع خودکار اتصال اعمال شده باشد و برای مدت مشخصی هیچ بستهٔ TPDU دریافت نشود (یا به هر دلیلی بسته‌های پوچ در میانهٔ راه از بین رفته باشند) اتصال به صورت خودکار خاتمه می‌یابد.[۱]

بیش از این به جزئیات نخواهیم پرداخت ولیکن تا اینجا باید مشخص شده باشد که خاتمه دادن به یک اتصال بدون از دست دادن داده، به آن سادگی هم که به نظر می‌رسد نیست!

۶-۲-۴ کنترل جریان و بافرسازی (Flow Control and Buffering)

پس از بررسی جزئیات روشهای برقراری و خاتمهٔ یک اتصال، اجازه بدهید چگونگی مدیریت یک اتصال را در حین کار، مطالعه نماییم. یکی از موارد بسیار مهم «کنترل جریان» (Flow Control) است. اگرچه از بسیاری جهات، مسائل مربوط به کنترل جریان در لایهٔ انتقال با مکانیزمهای کنترل جریان در لایهٔ پیوند داده مشابه هستند ولیکن از جهات دیگر دارای تفاوت هستند. تشابه عمده در هر دو لایه آن است که برای جلوگیری از پیشی گرفتن فرستندهٔ سریع از گیرندهٔ کُند و از دست رفتن داده‌ها در اثر عدم هماهنگی سرعت ارسال و دریافت، باید از پروتکل «پنجرهٔ لغزان» (Sliding Window) یا روشی مشابه استفاده شود. تفاوت اساسی این دو لایه آن است که یک مسیریاب معمولاً دارای تعداد کمی خط ارتباطی است در حالی که یک ماشین میزبان (Host) می‌تواند بطور همزمان تعداد بی‌شماری اتصال مختلف برقرار کند. این تفاوت موجب می‌شود که نتوان استراتژیهای بافرسازی پیاده شده در لایهٔ پیوند داده را به لایهٔ انتقال نیز تعمیم داده و اعمال نمود.

در پروتکلهای پیوند داده که در فصل سوم بررسی شدند، فریمها هم در مسیریاب فرستنده و هم در مسیریاب گیرنده، بافر می‌شدند. به عنوان مثال در پروتکل ششم، به ازای هر خط ارتباطی هم فرستنده و هم گیرنده به تعداد

---

۱. به عبارت ساده‌تر اگر هر یک از طرفین ارتباط برای مدت معینی از طرف مقابل خود بسته‌ای دریافت نکند، اتصال را قطع خواهد کرد لذا طرفین موظفند حتی اگر برای لحظاتی داده برای ارسال نداشته باشند، از خود علائم حیاتی نشان داده و بسته‌های پوچ و بدون داده (Dummy) ارسال نمایند تا قانون قطع خودکار اتصالات نیمه باز به درستی کار کند. -م

MAX_SEQ+1 بافر اختصاصی نیاز داشتند که از این تعداد نصفی برای فریمهای ورودی و نصف دیگر برای فریمهای خروجی در نظر گرفته می‌شد. برای ماشینی که حداکثر تعداد اتصالات آن مثلاً ۶۴ تاست و شماره‌های ترتیب بسته‌ها ۴ بیتی هستند، این پروتکل به ۱۰۲۴ بافر نیازمند است.

در لایهٔ پیوند داده، فرستنده فریمهای ارسالی خود را بافر می‌کند تا در صورت نیاز به ارسال مجدد، آنها را در اختیار داشته باشد. اگر زیرشبکه فقط سرویس دیتاگرام عرضه کرده باشد، «واحد انتقال» (Transport Entity) نیز به دلیل مشابه باید بسته‌های ارسالی خود را در بافر نگاهداری کند. در لایهٔ انتقال اگر گیرنده بداند که فرستنده تمام بسته‌های TPDU ارسالی خود را تا زمانی که دریافت آنها تصدیق (Ack) نشود در بافر نگهداری می‌کند، می‌تواند بر حسب شرایط برای اتصالات خاصی بافر ورودی اختصاصی در نظر نگیرد. (یا برعکس به هر اتصال، بافر ورودی و خروجی مجزا اختصاص بدهد.) به عنوان مثال در لایهٔ انتقال، گیرنده می‌تواند از یک فضای بزرگ و مرکزی به عنوان بافر ورودی برای تمام اتصالات استفاده کند. بدین نحو وقتی یک TPDU وارد می‌شود، گیرنده سعی می‌کند به صورت پویا از این بافر فضایی را به آن اختصاص بدهد. اگر چنین فضایی موجود بود بستهٔ TPDU را می‌پذیرد و در غیر این صورت آن را حذف می‌نماید. از آنجایی که فرستندهٔ بسته، برای ارسال مجدد بسته‌هایی که در زیرشبکه از دست می‌روند، آمادگی دارد فلذا حذف TPDU توسط گیرنده مشکلی ایجاد نخواهد کرد، اگرچه این کار منجر به از دست رفتن منابع [مثل پهنای باند] شبکه خواهد شد. فرستنده، ارسال مجدد بسته‌ها را آنقدر تکرار می‌کند تا بالاخره دریافت آن تصدیق شود.

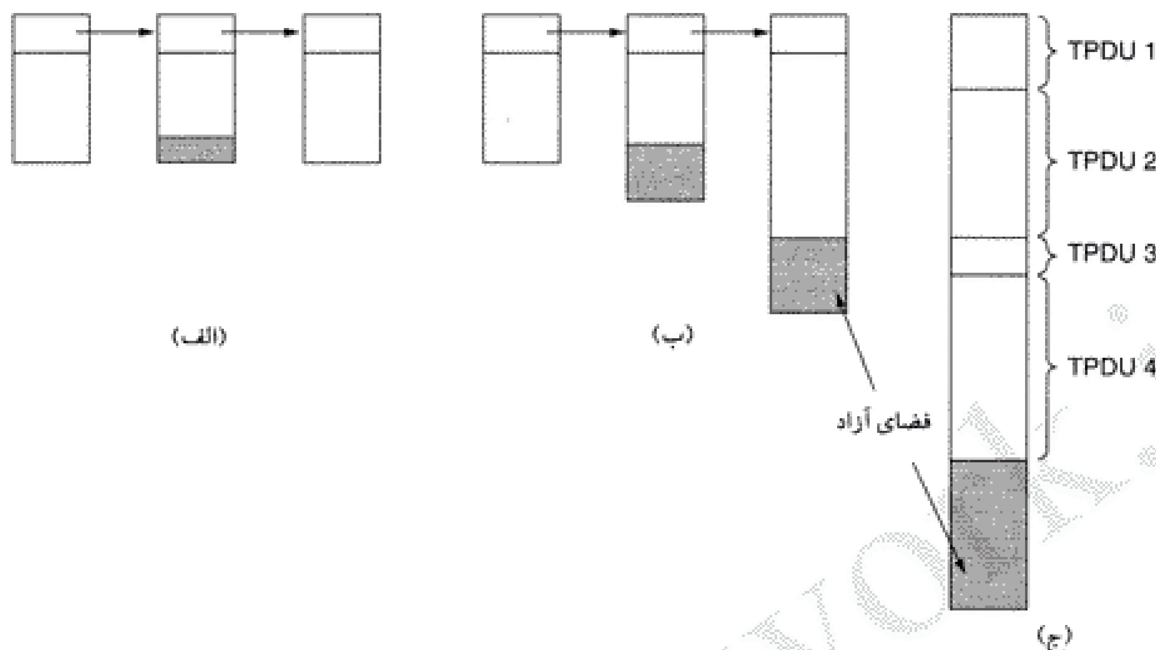
کوتاه سخن آن که، اگر سرویس شبکه غیرقابل اعتماد باشد، فرستنده باید تمام بسته‌های TPDU ارسالی خود را (همانند لایهٔ پیوند داده) بافر کند، ولیکن اگر سرویس شبکه قابل اعتماد باشد می‌توان از راهکارهای بینابینی بهره گرفت. بالاخص اگر گیرنده اطمینان داشته باشد که گیرنده همیشه فضای بافر کافی در اختیار دارد مجبور نیست که نسخه‌ای از TPDUهای ارسالی خود را نگه دارد ولیکن اگر گیرنده نتواند تضمین کند که بسته‌های TPDU ورودی را قطعاً خواهد پذیرفت، فرستنده در هر حال مجبور به بافر کردن آنهاست. در حالت دوم، فرستنده نمی‌تواند به پیغامهای اعلام وصول بسته‌ها (Ack) اعتماد کند چرا که پیغامهای اعلام وصول فقط بدین معناست که بسته‌ها رسیده‌اند نه آن که پذیرفته شده‌اند.[1] در ادامه باز هم به این نکتهٔ مهم باز خواهیم گشت.

حتی اگر گیرنده موافق بافر کردن بسته‌ها باشد باز هم سؤالی باقی می‌ماند و آن هم حجم بافر مورد نیاز آن است. اگر اغلب TPDUها اندازه‌ای تقریباً مساوی داشته باشند، طبیعی آنست که یک فضای حافظهٔ بزرگ با تعدادی بافر هم اندازه ایجاد نماییم (یک بسته در هر بافر). شکل ۶-۱۵-الف این مفهوم را به تصویر کشیده است. ولیکن اگر اندازهٔ بسته‌های TPDU تفاوت فاحش داشته باشند (مثلاً از چند بایت تایپ شده در ترمینال ـ مثلاً Telnet ـ گرفته تا هزاران کاراکتر در حین انتقال فایل)، استفاده از بافرهایی با طول ثابت مشکل‌ساز خواهد شد: از یک طرف اگر اندازهٔ بافرها را معادل با اندازهٔ بزرگترین بستهٔ TPDU در نظر بگیریم، برای بسته‌های کوچک فضای حافظه هدر خواهد رفت. از طرف دیگر، اگر اندازهٔ بافر کمتر از حداکثر طول بسته‌های TPDU انتخاب شود برای بسته‌های بزرگ به چندین بافر نیاز خواهد بود و به پیچیدگی روش می‌انجامد.

راهکار دیگر برای حل مسئلهٔ طول بافر آنست که از بافرهایی با طول متغیر بهره بگیریم. این مفهوم در شکل ۶-۱۵-ب نشان داده شده است. مزیت این روش، استفادهٔ مفید از فضای حافظه است ولیکن به بهای پیچیدگی مکانیزمهای مدیریت بافر تمام می‌شود. راه حل دیگر آن است که به ازای هر اتصال، یک بافر چرخه‌ای بزرگ (Circular Buffer) اختصاص بدهیم. (به شکل ۶-۱۵-ج دقت نمایید.) در این سیستم از حافظه، بخوبی استفاده می‌شود و در اتصالی که حجم مبادلهٔ بار سنگین است کارآیی خوبی دارد ولی برای اتصالی که حجم مبادلهٔ بار کمی

---

۱. ممکن است بسته‌ای به سلامت در ماشین گیرنده دریافت شود ولی پروسه‌ای که صاحب آن است به هر دلیل آن را نپذیرد. -م



شکل ۶-۱۵. (الف) بافرهای زنجیره‌ای با طول ثابت. (ب) بافرهای زنجیره‌ای با طول متغیر. (ج) یک بافر زنجیره‌ای بزرگ به ازای هر اتصال.

دارد ضعیف عمل می‌کند.

حالت بهینه و متعادل در بافرسازی بسته‌ها در سمت گیرنده و فرستنده به نوع ترافیکی بستگی دارد که از طریق هر اتصال [بین دو پروسه] مبادله می‌شود. برای ترافیکهای انفجاری (Bursty) که به پهنای باند کمی احتیاج دارند (مثل داده‌هایی که از طریق یک ترمینال محاوره‌ای تولید و ارسال می‌شود) بهتر آن است که هیچ بافری از قبل اختصاص داده نشود و در عوض بافر مورد نیاز در هر دو سمت (گیرنده و فرستنده) به صورت پویا و برحسب نیاز تخصیص داده شود. (Dynamic Allocation) از آنجایی که در تخصیص پویا، فرستنده اطمینان ندارد که گیرنده قادر به تخصیص حافظه لازم خواهد بود لذا فرستنده باید نسخه‌ای از هر بستهٔ TPDU ارسالی خود را (مادامی که دریافت آنها تصدیق نشده)، در بافر نگاه دارد. از طرف دیگر برای انتقال فایل یا هر کاربرد دیگری که به پهنای باند بالا نیاز دارد بهتر آن است که گیرنده ، یک پنجرهٔ کامل از بافرها را از قبل رزرو کند تا داده‌ها بتوانند با حداکثر نرخ ممکن جریان داشته باشند.[1] بنابراین برای ترافیک انفجاری ولی با پهنای باند کم بهتر است که فرآیند بافرسازی به نحو جدی در سمت فرستنده انجام شود ولی برای ترافیک یکنواخت با پهنای باند بالا بافرسازی در سمت گیرنده مفیدتر خواهد بود.

وقتی اتصالی باز یا بسته می‌شود یا الگوی ترافیک تغییر می‌کند، گیرنده و فرستنده ملزم به تنظیم خودکار و پویای فضای بافرهای خود هستند. در نتیجه پروتکل لایهٔ انتقال باید این امکان را فراهم کند که ماشین فرستنده، فضای بافر مورد نیاز را در ماشین سمت مقابل خود، سفارش داده و رزرو نماید. بافرها را می‌توان برای هر اتصال بطور مجزا اختصاص داد یا آن که بافرها به صورت یکجا و برای کل اتصالاتی که بین دو ماشین برقرار می‌شود، رزرو گردد. همچنین در سمت مقابل، گیرنده‌ای که از وضعیت بافر خود آگاه است (ولی حجم ترافیک عرضه شده توسط ماشین مقابل را نمی‌داند) می‌تواند به فرستنده اعلام کند که مثلاً: «من به تعداد x بافر برای شما کنار گذاشته‌ام».

---

۱. تا بدلیل مشکلاتی که در حین تخصیص پویای حافظه رخ می‌دهد جریان داده‌ها ناگزیر به قطع موقت نشود. -م

هرگاه تعداد اتصالات باز افزایش یابد ممکن است هر یک از طرفین بدلیل محدودیت حافظه مجبور به کاهش حجم بافر تخصیص یافته شوند، لذا پروتکل لایهٔ انتقال باید از چنین قابلیتی برخوردار باشد.

روش عمومی و عقلانی مدیریت پویای بافرها آن است که مسئله بافرسازی را از مسئلهٔ اعلام وصول بسته‌ها (Acknowledgements) تفکیک نماییم.[1] (برخلاف پروتکل پنجرهٔ لغزان که در فصل سوم تشریح شد.) در نتیجه مدیریت پویای بافرها مستلزم داشتن پنجره‌ای با طول متغیر است. در ابتدا فرستنده براساس پیش‌بینی‌های اولیه میزان بافر مورد نیاز خود را اعلام می‌کند. گیرنده تا حدی که برایش مقدور است حجم خواسته شده را اختصاص می‌دهد. هر وقت فرستنده یک TPDU ارسال کرد، حجم آن را از میزان فضای اختصاص داده شده کم می‌کند و هر گاه میزان این فضا به صفر رسید، ارسال را متوقف می‌نماید. گیرنده نیز اعلام وصول بسته‌ها (Ackها) و همچنین فضای بافر موجود خود را در ترافیک برگشتی (Reverse Traffic) اعلام می‌کند.

شکل ۶–۱۶ مثالی از مدیریت پویای پنجره را در زیرشبکه‌ای دیتاگرام نشان می‌دهد که در آن شماره‌های ترتیب ۴ بیتی هستند. فرض کنید که اطلاعات لازم برای تخصیص بافر در بسته‌های جداگانهٔ TPDU ارسال گردد و در ترافیک برگشتی، جاسازی (Piggyback) نشود. در ابتدا، A هشت بافر تقاضا می‌دهد ولی فقط با چهار بافر موافقت می‌شود. A سه بستهٔ TPDU ارسال می‌کند که سومین آنها از بین می‌رود. [به سطرهای ۱ تا ۵ از شکل ۶–۱۶ نگاه کنید.] در بستهٔ TPDU ششم دریافت بسته‌های ۰ و ۱ تأیید می‌شود و A اجازه می‌یابد تا بافرهای متعلق به این بسته‌ها را آزاد کند [سطر ششم از شکل]. عبارت <ack=1,buf=3> بدین معناست که بسته‌ها تا شمارهٔ ۱ به درستی دریافت شده‌اند و فرستنده باید بسته‌های از ۲ به بعد را ارسال کند؛ در ضمن buf=3 تعداد بافرها را مشخص می‌کند و طبعاً فرستنده مجاز به ارسال حداکثر سه بستهٔ TPDU است (یعنی بسته‌های ۲، ۳، ۴). A می‌داند که بستهٔ شمارهٔ ۲ را قبلاً ارسال کرده، فلذا گمان می‌کند که باید بسته‌های ۳ و ۴ را بفرستد و این کار را انجام می‌دهد. در این نقطه A متوقف می‌شود و صبر می‌کند تا فضای بافر طرف مقابل آزاد شده و به او اعلام شود. در سطر نهم از شکل، می‌بینیم که در اثر انقضای مهلت، بستهٔ دوم (که قبلاً از بین رفته) از نو ارسال شده است. (البته ممکن بود انقضای مهلت تایمر زمانی رخ بدهد که به دلیل عدم وجود فضای بافر، فرستنده متوقف شده باشد.) در سطر دهم از شکل، B دریافت تمام بسته‌های TPDU تا شماره ۴ را اعلام می‌کند [یعنی بسته‌های ۲ و ۳ و ۴ به سلامت رسیده‌اند] ولیکن کماکان به A اجازهٔ ادامهٔ ارسال را نمی‌دهد [چون با اعلام buf=0 فضای بافر خود را صفر گزارش کرده است و A اجازه ندارد چیزی بفرستد.] به خاطر داشته باشید که چنین وضعیتی برای «پروتکل پنجرهٔ ثابت» (Fixed Window Protocol) هرگز پیش نمی‌آید [چرا که در آنجا اعلام وصول فریمها به منزلهٔ آزاد شدن فضای بافر نیز تلقی می‌شود.] در سطر یازدهم، یک بستهٔ TPDU از B به A ارسال شده و ضمن اعلام وجود یک بافر خالی، به A اجازهٔ ادامهٔ ارسال می‌دهد.

روشهای تخصیص بافر همانند روش مثال فوق، مستعد بروز مشکلاتی هستند که در اثر از بین رفتن بسته‌های کنترلی (Control TPDU) در زیرشبکه‌های دیتاگرام، رخ می‌دهند. به سطر شانزدهم از شکل ۶–۱۶ دقت نمائید: B چهار بافر آزاد برای A اختصاص داده و آن را در یک بستهٔ کنترلی به سوی A فرستاده ولیکن این بسته از بین رفته است. از آنجایی که بسته‌های کنترلی شماره‌گذاری نشده‌اند و هیچ تایمری جهت ارسال مجدد ندارند فلذا A

---

۱. یعنی دریافت Ack فقط بیانگر آنست که داده‌های ارسالی بسلامت رسیده‌اند ولی نباید بدین معنا تلقی شود که فرستنده حق ارسال بسته‌های بعدی را دارد چراکه ممکن است بسته‌ها هنوز تحویل پروسهٔ کاربردی نشده باشد و هیچ بافر خالی دیگر برای دریافت بستهٔ بعدی موجود نباشد. –م

	A	پیام	B	توضیح
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	•••	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	•••	<ack = 6, buf = 4>	←	Potential deadlock

شکل ۶-۱۶. تخصیص بافر بصورت پویا. (فلشها جهت ارسال را مشخص می‌کنند. علامت ... نشانگر یک TPDU از دست رفته، می‌باشد.)

در یک بن‌بست (Deadlock) قرار می‌گیرد.[1] برای پیشگیری از بروز چنین وضعیتی، هر ماشین باید بطور متناوب بسته‌های کنترلی TPDU (شامل وضعیت بافرها و شمارهٔ Ack) را بر روی هر اتصال بیکار ارسال نماید. بدین نحو، دیر یا زود بن‌بست شکسته خواهد شد.

تا اینجا بطور ضمنی فرض کرده‌ایم که تنها محدودیتی که بر روی نرخ ارسال داده‌های فرستنده تحمیل می‌شود، فضای بافر موجود در گیرنده است.[2] در حالی که با روند رو به کاهش قیمت حافظه، امکان آن فراهم شده که هر ماشین میزبان به حجم بسیار انبوه حافظه مجهز گردد و بدین ترتیب کمبود حافظه به ندرت رخ داده و مشکل بافرها حل می‌شود.

وقتی فضای بافر محدودیتی بر روی حداکثر میزان جریان اعمال نکند گلوگاه دیگری بروز می‌کند: **«ظرفیت حمل زیرشبکه»**[3]. اگر مسیریابهای مجاور قادر به مبادلهٔ حداکثر x بسته در هر ثانیه باشند و در مجموع k مسیر مستقل و مجزا بین یک زوج ماشین وجود داشته باشد، این دو ماشین به هیچ روشی نمی‌توانند بیش از $k.x$ بستهٔ TPDU (در هر ثانیه) مبادله نمایند و فضای بافر موجود در هر یک از طرفین در این مقدار تاثیری ندارد. اگر فرستنده بار سنگینی را به شبکه تزریق کند (به عبارتی بسته‌های TPDU را با سرعتی بیش از $k.x$ TPDU/sec ارسال کند)، زیرشبکه با ازدحام مواجه می‌شود، زیرا قادر نیست بسته‌ها را با همان سرعتی که دریافت می‌کند، تحویل بدهد و بدین ترتیب بخشی از آنها از بین می‌روند.

---

۱. زیرا B با خود می‌اندیشد که چون به A گفته که بافر خالی دارد احتمالاً A داده‌ای برای ارسال نداشته و A هم با خود می‌اندیشد که شاید بافرهای B هنوز خالی نشده است و بدین نحو یک دور باطل ایجاد می‌شود. -م

۲. ازین به بعد فرض بر آنست که گیرنده به طور متناوب فضای بافر خود را به فرستنده اعلام می‌کند و فرستنده ملزم به ارسال داده متناسب با این فضا است. -م

۳. Network's Carrying Capacity

در اینجا به مکانیزمی نیاز داریم که مبتنی بر ظرفیت حمل زیرشبکه، جریان بسته‌ها را کنترل کند نه براساس ظرفیت بافر در گیرنده. روشن است که مکانیزم کنترل جریان بایستی در سمت فرستنده اعمال گردد تا بسته‌های بیهوده‌ای که دریافت آنها تصدیق نشده، متوالیاً ارسال و در زیرشبکه سرگردان نشوند.[۱] «بلسنس» (Belsnes) در سال ۱۹۷۵ با استفاده از «پروتکل پنجرهٔ لغزان» (Sliding Window) روشی پیشنهاد کرد که در آن فرستنده برای کنترل جریان، باید اندازهٔ پنجرهٔ خود را (به صورت پویا) به نحوی تنظیم کند که با ظرفیت حمل شبکه متناسب باشد. اگر شبکه بتواند حداکثر c TPDU/sec را حمل نماید و «**زمان گردش**» (شامل زمان انتقال، تأخیر انتشار، تأخیر انتظار در صف، زمان پردازش در گیرنده و زمان برگشت پیام اعلام وصول ـAck) نیز r ثانیه فرض شود، ظرفیت پنجرهٔ فرستنده باید معادل $c.r$ باشد. اگر پنجره‌ای با این اندازه انتخاب شود، فرستنده می‌تواند در شرایطی کاملاً عادی و به مثابهٔ یک خط لوله (Pipeline) عمل کند. [بدون آن که بسته‌ای در اثر ازدحام از دست برود.] یک کاهش کوچک در کارآیی شبکه می‌تواند موجب توقف او شود.

برای آن که بتوان اندازهٔ پنجره را به صورت متناوب و خودکار تنظیم کرد، فرستنده می‌تواند بر مقدار این دو پارامتر [یعنی ظرفیت حمل شبکه و زمان گردش] نظارت داشته باشد و اندازهٔ پنجرهٔ خود را براساس آنها تعیین نماید. برای محاسبهٔ ظرفیت حمل شبکه می‌توان به سادگی تعداد بسته‌هایی را که در دورهٔ زمانی مشخص، ارسال و دریافت آنها تصدیق می‌شود، شمرد و حاصل را بر طول زمان تقسیم کرد. در خلال زمان محاسبه، فرستنده باید بسته‌های خود را با حداکثر نرخ ممکن ارسال نماید تا مطمئن شود آنچه که برگشت بسته‌های اعلام وصول (Ack) را محدود کرده ظرفیت حمل شبکه است نه سرعت پایین خودش. زمان ارسال یک بستهٔ TPDU و برگشت پیغام وصول آن (Ack) را می‌توان به دقت اندازه‌گیری کرد. از آنجایی که ظرفیت فعلی شبکه متغیر با زمان است، اندازهٔ پنجره نیز باید به دفعات تنظیم شود تا هر گونه تغییر در ظرفیت شبکه را دنبال کرده و اندازهٔ پنجره را با آن تطبیق بدهد. بعداً خواهیم دید که اینترنت از روشی شبیه به همین الگو استفاده می‌کند.

۶-۲-۵ مالتی‌پلکسینگ (تسهیم)

مالتی‌پلکس کردن چندین محاورهٔ همزمان (Conversation) بر روی اتصالات، مدارات مجازی یا لینکهای فیزیکی، نقش بسیار مهمی در لایه‌های مختلف معماری یک شبکه ایفاء می‌کند.[۲] در لایهٔ انتقال از چندین جهت به عمل مالتی‌پلکس نیاز است. به عنوان مثال اگر یک ماشین میزبان فقط دارای یک آدرس در شبکه باشد[۳]، تمام اتصالات برقرار شده در لایهٔ انتقال مجبور به مالتی‌پلکسینگ هستند. یعنی به راهکاری نیاز است تا هر گاه یک TPDU وارد می‌شود بتوان پروسهٔ تحویل‌گیرندهٔ آنرا، مشخص کرد. به این وضعیت «**مالتی پلکس روبه‌بالا**» (Upward Multiplexing) گفته می‌شود و شمائی از آن در شکل ۶-۱۷-الف نشان داده شده است. در این شکل چهار اتصال ایجاد شده در لایهٔ انتقال، همگی از اتصال مشترکی در لایهٔ شبکه (مثلاً آدرس IP مشترک) بهره گرفته‌اند.[۴]

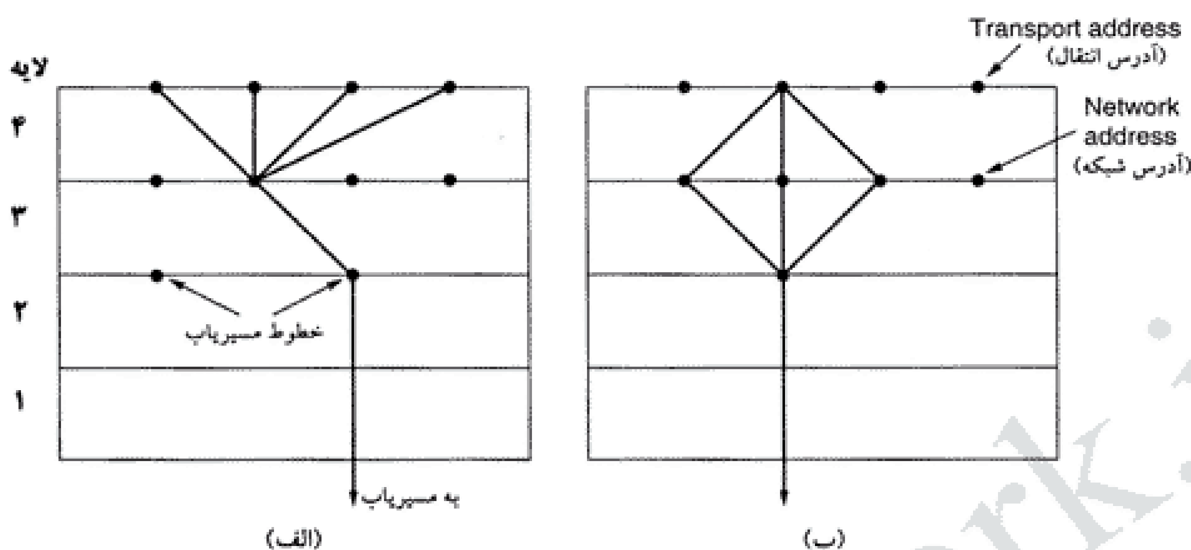
---

۱. بعبارت دیگر اگر ظرفیت حمل شبکه محدود باشد فقط فرستنده می‌تواند جریان ارسال داده‌های خود را متناسب با این ظرفیت تنظیم کند؛ بنابراین هر مکانیزمی بدین منظور، فقط بر روی فرستنده قابل اجراست. ـم

۲. به عبارت دیگر عمل مالتی‌پلکس را می‌توان مکانیزمی جهت استفادهٔ مشترک چندین پروسه از یک لینک واحد، مدار مجازی واحد یا آدرس مشترک تعبیر کرد. ـم

۳. عموماً اینگونه است و اغلب ماشینهای اینترنت فقط یک آدرس IP دارند. ـم

۴. به عبارت دیگر تمام داده‌های این چهار پروسه وقتی به لایهٔ شبکه می‌رسند درون بسته‌هایی قرار می‌گیرند که آدرس مبداء همهٔ آنها یکسان است و همگی نهایتاً بر روی یک لینک مشترک ارسال می‌شوند و باید بنحوی در لایهٔ انتقال از یکدیگر تفکیک و به پروسه‌های متناظر خود تحویل داده شوند. فرآیند تفکیک بسته‌ها بین پروسه‌ها در لایهٔ انتقال، «مالتی‌پلکس روبه‌بالا» نام دارد.



شکل ۶-۱۷. (الف) مالتی‌پلکس روبه بالا. (ب) مالتی‌پلکس رو به پایین.

مالتی پلکس در لایهٔ انتقال، از جهت دیگری نیز می‌تواند مفید واقع شود. بـه عـنوان مـثال فـرض کـنید در زیرشبکه‌ای که از درون مبتنی بر مدار مجازی (Virtual Circuit) است، بر روی حداکثر نرخ ارسال هـر مـدار مجازی، محدودیت گذاشته شده است. اگر کاربر به پهنای باند بیشتری نسبت به نرخ حداکثر هر مدار مجازی نیاز داشته باشد، یک راهکار آن است که چندین اتصال مدار مجازی همزمان در شبکه ایجاد شود و ترافیک داده‌های یک پروسه به نوبت و چرخشی (Round Robin) بر روی این مدارات مجازی توزیع شود. این مـفهوم کـه در شکل ۶-۱۷-ب به تصویر کشیده شده است، «**مالتی‌پلکس روبه‌پایین**» نام دارد. با داشتن k اتصال باز در سطح شبکه (بعبارت دیگر k مدار مجازی همزمان)، پهنای باند مؤثر با ضریب k افزایش می‌یابد. مثالی از مالتی‌پلکس روبه‌پایین را می‌توان کاربران خانگی عنوان کرد که از خط ISDN بهره می‌گیرند. این خط دو اتصال 64kbps دو طرفه را در اختیار می‌گذارد. با استفادهٔ همزمان از این دو اتصال برای وصل به یک شرکت ارائه دهـندهٔ خـدمات اینترنت و توزیع ترافیک بر روی آنها، پهنای باند مؤثر 128kbps حاصل می‌شود.

۶-۲-۶ جبران از کارافتادگی (Crash Recovery)

اگر ماشینهای میزبان یا مسیریابها مستعد خرابی و از کارافتادگی باشند، موضوع احیاء و برگرداندن آنها به فعالیت طبیعی، مسئلهٔ مهمی است. اگر «**واحدهای انتقال**» (Transport Entities) بطور کلی در درون ماشینهای میزبان مستقر باشند، از کارافتادگی شبکه و مسیریاب به سادگی اصلاح و جبران خواهد شد. اگر چنین شبکه‌ای خدمات دیتاگرام عرضه نماید، «واحدهای انتقال» همیشه آمادگی از بین رفتن بسته‌های TPDU را دارند و روش برخورد با چنین مشکلی را می‌دانند.[1] اگر لایهٔ شبکه خدمات مدار مجازی عرضه کرده باشد، از دست رفتن یک مدار مجازی بدین نحو جبران می‌شود که مدار مجازی جدیدی برقرار شده و از واحد انتقال در ماشین راه دور سؤال می‌گردد که چه TPDUهایی را دریافت کرده و کدام را دریافت ننموده است؛ آنهایی که دریافت نشده‌اند از نو ارسال می‌شوند.

مسئله دردسرآفرین، آنست که ماشینهای میزبان را چگونه پس از خرابی به فعالیت طبیعی بازگردانیم؟ برای آن که دشوار بودن این عمل را نشان بدهیم فرض کنید که یک ماشین میزبان در حال ارسال یک فایل طولانی برای

۱. بعبارتی از بین رفتن بسته‌ها در شبکه‌های دیتاگرام امری طبیعی است و نیاز به مکانیزمی اضافی ندارد، لذا خرابی یک کانال یا از کارافتادن موقت مسیریابهای میانی مشکلی برای لایهٔ انتقال ایجاد نمی‌کند. -م

ماشین میزبان دیگر (مثلاً سرویس‌دهندهٔ فایل) به روش «**توقف و انتظار**» (Stop & Wait) است. [یعنی یک قطعه از فایل ارسال شده و منتظر اعلام وصول آن می‌ماند.] لایهٔ انتقال در ماشین سرویس‌دهنده، بسته‌های TPDU را یکی‌یکی به پروسهٔ کاربردی مربوطه تحویل می‌دهد. در اثنای کار، سرویس‌دهنده از کار می‌افتد؛ پس از راه‌اندازی مجدد، وقتی این ماشین فعالیت طبیعی خود را از سر می‌گیرد، جداول او مقداردهی اولیه می‌شوند فلذا او نمی‌داند که دقیقاً در کجای کار این خرابی اتفاق افتاده و طبعاً همه چیز باید از نو تکرار شود.

برای آن که بتوان وضعیت را به حالت قبلی برگرداند، سرویس‌دهنده می‌تواند یک بستهٔ TPDU را به روش پخش فراگیر (Broadcast) برای بقیهٔ ماشینهای میزبان بفرستد و با اعلام آن که از کار افتاده بوده از تمام ماشینهای مشتری (Clients) درخواست کند که وضعیت تمام اتصالات باز خود را [که با او برقرار کرده بوده‌اند]، اعلام نمایند. هر مشتری ممکن است یکی از این دو وضعیت را داشته باشد: منتظر یک TPDU باشد (وضعیت S1) منتظر هیچ TPDU نباشد (وضعیت S0). براساس این اطلاعات وضعیت، مشتری می‌تواند در خصوص ارسال مجدد TPDUهای اخیر تصمیم بگیرد.

در نگاه اول این فرآیند ساده به نظر می‌رسد: مشتری باید بسته‌های TPDU اعلام وصول نشده را از نو ارسال نماید (یعنی وضعیت S1). ولیکن بررسی موشکافانهٔ این روش، مشکلات آن را آشکار می‌کند. به عنوان مثال وضعیتی را در نظر بگیرید که واحد انتقال در ماشین سرویس‌دهنده، پیام اعلام وصول (Ack) یک بستهٔ TPDU را ارسال کرده و محتوای آن را به برنامهٔ کاربردی تحویل داده است. نوشتن یک TPDU در یک استریم و سپس ارسال پیغام اعلام وصول دو رخداد کاملاً مجزا است و نمی‌تواند بطور همزمان انجام شود. اگر خرابی ماشین دقیقاً زمانی رخ بدهد که پیغام اعلام وصول بسته، ارسال شده ولی محتوای آن بسته هنوز در استریم متعلق به برنامهٔ کاربردی نوشته نشده ، مشتری پیغام اعلام وصول بسته را دریافت می‌کند و طبعاً در وضعیت S0 قرار می‌گیرد ولیکن محتوای بسته حقیقتاً به برنامهٔ کاربردی نرسیده و ماشین مشتری آن را از نو ارسال نخواهد کرد چرا که به غلط فکر می‌کند که این TPDU دریافت شده است. چنین تصمیمی منجر به از دست رفتن یک TPDU خواهد شد.

در اینجا ممکن است با خود بیندیشید که: «این مشکل را براحتی می‌توان حل کرد. کل کاری که باید انجام شود آنست که واحد انتقال (Transport Entity) بازنویسی و اصلاح شود تا اول بسته را بنویسد و سپس آن را اعلام وصول کند!» ولیکن تجسم کنید که ابتدا عمل نوشتن در استریم برنامهٔ کاربردی انجام شود ولی دقیقاً قبل از ارسال پیغام اعلام وصول، ماشین از کار بیفتد. این رخداد منجر به تولید بسته‌های TPDU تکراری شده و چون تکراری بودن آنها کشف نمی‌شود در استریم خروجی پروسهٔ کاربردی سرویس‌دهنده نوشته خواهد شد.

فارغ از آنکه برنامهٔ سرویس‌دهنده و مشتری چگونه برنامه‌نویسی شده‌اند همیشه وضعیتی وجود دارد که پروتکل در حین برگشت به حالت طبیعی (پس از خرابی ماشین) با شکست جدی مواجه می‌شود. سرویس‌دهنده را می‌توان به یکی از دو روش ذیل برنامه‌نویسی کرد: اول پیغام اعلام وصول بسته را بفرستد یا اول محتوای بسته را در استریم پروسهٔ کاربردی بنویسد. برنامهٔ مشتری را می‌توان به چهار روش نوشت: (۱) همیشه آخرین TPDU ارسالی خود را مجدداً بفرستد. (۲) هیچگاه آخرین بستهٔ TPDU را نفرستد. (۳) فقط وقتی بستهٔ آخر را مجدداً ارسال کند که در وضعیت S0 باشد. (۴) بستهٔ آخر را فقط وقتی از نو ارسال کند که در وضعیت S1 باشد. ترکیب اینها هشت حالت مختلف را پدید می‌آورد ولیکن به گونه‌ای که خواهیم دید برای هر ترکیب، مجموعه‌ای از رخدادها منجر به شکست پروتکل می‌شود.

در سرویس‌دهنده وقوع سه رخداد محتمل است: (۱) ارسال پیام Ack (A) (۲) نوشتن در پروسهٔ خروجی (W) (۳) از کارافتادگی (C). این سه رخداد می‌تواند به شش ترتیب مختلف اتفاق بیفتد: **AC(W)** ، **AWC** ،

C(AW)، C(WA)، WAC و WC(A). پرانتزها نمایانگر آن هستند که وقتی سیستم از کار می‌افتد طبعاً مراحل بعدی که درون پرانتز نشان داده شده‌اند، نمی‌تواند ادامه پیدا کند. (یعنی وقتی سیستم خراب شد، کار تمام است و رخدادهای درون پرانتز فرصت وقوع پیدا نمی‌کنند.) شکل ۶-۱۸ هشت ترکیب مختلف ناشی از عملکرد سرویس‌دهنده و مشتری و ترکیبات معتبر آنها نشان داده شده‌اند. دقت کنید که در هر استراتژی، دنباله‌ای از رخدادها می‌تواند منجر به شکست پروتکل شود. به عنوان مثال، اگر ماشین مشتری، همیشه آخرین بستهٔ ارسالی خود را ارسال کند، استراتژی AWC[1] منجر می‌شود که بسته‌ای تکراری و غیرقابل تشخیص، دریافت و تحویل پروسه شود، حتی اگر دو رویداد دیگر [یعنی AW] به درستی انجام گردد.

استراتژیهای بکار رفته در ماشین فرستنده	استراتژیهای بکار رفته در ماشین گیرنده: First ACK, then write			First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = پروتکل بدستی عمل می‌کند.
DUP = پروتکل یک پیام تکراری تولید می‌کند.
LOST = پروتکل پیامی را از دست می‌دهد.

شکل ۶-۱۸. ترکیبات مختلف استراتژیهای اتخاذ شده در سرویس‌دهنده و مشتری.

پیچیده و هوشمندتر کردن پروتکل کمک چندانی به حل مشکل نمی‌کند. حتی اگر مشتری و سرویس‌دهنده، قبل از آنکه سرویس‌دهنده تلاش کند داده‌ها را به پروسهٔ کاربردی تحویل بدهد (یا به عبارت دیگر آنها را بنویسد) چندین بستهٔ TPDU مبادله کنند تا مشتری بداند چه اتفاقی در حال وقوع است، باز هم در صورت از کار افتادن سرویس‌دهنده و برگشت به حالت طبیعی، مشتری نمی‌تواند بفهمد که وقوع خرابی قبل از نوشتن بسته‌ها بوده یا بعد از آن. بنابراین ناگزیر به پذیرش این نتیجه هستیم: در شرایطی که رخدادها بطور همزمان اتفاق نمی‌افتند[2] از کار افتادن ماشین میزبان و بازگشت آن به حالت طبیعی را نمی‌توان بی‌سر و صدا اصلاح کرد و از دید لایه‌های بالاتر مخفی نخواهد ماند.[3]

اگر بخواهیم نتیجه‌گیری فوق را تعمیم بدهیم می‌توان آن را بدین نحو بیان کرد که **«اگر لایهٔ N دچار از کار افتادگی شود، بازیابی و بازگرداندن آن به شرایط طبیعی، تنها در لایهٔ N+1 مقدور است و آن هم مشروط به آنکه لایهٔ بالایی اطلاعات کافی از وضعیت فعلی نگه داشته باشد.»** در بالا اشاره شد که هر گونه خرابی در لایهٔ شبکه می‌تواند توسط لایهٔ انتقال اصلاح و جبران شود به شرط آن که هر یک از طرفین یک اتصال، وضعیت فعلی خود و دیگری را نگه داشته باشند.

---

۱. AWC یعنی ابتدا اعلام وصول بسته، سپس تحویل داده‌ها به پروسه و سپس از کار افتادن ماشین -م

۲. یعنی اول اعلام وصول و بعد تحویل داده‌ها به پروسه انجام می‌شود یا بالعکس ولی نه **همزمان** -م

۳. بدون تمهیدات برنامه‌های کاربری در لایهٔ کاربرد، داده‌هایی که قبل از خرابی ماشین دریافت شده‌اند از درجهٔ اعتبار ساقط است. -م

این مسئله ما را متوجه معنای حقیقی موضوعی می‌کند که اصطلاحاً با عنوان «**تصدیق دریافت داده‌ها به صورت انتهابه‌انتها**» (End-to-End Acknowledgement) معرفی می‌شود. اصولاً پروتکل لایهٔ انتقال «انتهابه‌انتها» است و همانند لایه‌های زیرین به صورت «زنجیره‌ای» عمل نمی‌کند.[1] حال وضعیتی را مدّنظر قرار بدهید که یک کاربر تقاضایی را بر روی یک پایگاه دادهٔ راه دور اعمال می‌کند. فرض کنید واحد انتقال (Transport Entity) به گونه‌ای برنامه‌نویسی شده که ابتدا بسته‌های TPDU را به لایهٔ بالاتر تحویل بدهد و بعد از آن پیغام تصدیق دریافت (Ack) بفرستد. در چنین حالتی حتی بازگشت پیغام Ack به ماشین کاربر، الزاماً به معنای آن نیست که این درخواست واقعاً بر روی بانک اطلاعاتی اعمال و بهنگام‌سازی شده است. یک مکانیزم «تصدیق انتهابه‌انتها» که در آن دریافت Ack به معنای انجام صددرصد کار و عدم دریافت آن به معنای انجام نشدن کار باشد، در عمل دست نیافتنی است. این نکته به تفصیل در مرجع (Saltzer et al. 1984) بحث شده است.

۶ـ۳ یک پروتکل سادهٔ انتقال

برای تبیین ایده‌هایی که تا اینجا بررسی شدند، در این بخش به مطالعهٔ مبسوط یک مثال عملی از لایهٔ انتقال می‌پردازیم. خدمات اولیه ارائه شده همان توابع اولیهٔ (Primitives) اتصال‌گرا هستند که در شکل ۶-۲ نشان داده شده‌اند. انتخاب این توابع و عملکرد اتصال‌گرای اولیه، مثال ما را بسیار شبیه به پروتکل TCP (ولی ساده‌تر از آن) خواهد نمود.

۶ـ۳ـ۱ توابع اولیهٔ ارائه خدمات در مثال فوق

اولین مسئله‌ای که با آن مواجهیم تشریح صحیح و دقیق «عملکردهای اولیهٔ انتقال» است. عمل CONNECT ساده است: ما یک تابع کتابخانه‌ای به نام ***connect*** خواهیم داشت که برای برقراری یک اتصال، می‌توان آنرا با پارامترهای مناسب فراخوانی کرد. این پارامترها عبارتند از شناسهٔ TSAP مبداء (محلی) و شناسهٔ TSAP مقصد (راه دور). پس از فراخوانی، پروسهٔ صدا زننده متوقف می‌شود (به عبارت دیگر معلق می‌گردد) تا واحد انتقال برای برقراری اتصال اقدام کند. اگر برقراری اتصال انجام شود، پروسهٔ صدازننده از حالت توقف خارج شده و می‌تواند شروع به ارسال داده بنماید.

هر گاه پروسه‌ای در سمت سرویس‌دهنده بخواهد قادر به پذیرش تقاضاهای ارتباط باشد، تابع ***listen*** را فراخوانی کرده و TSAP مشخصی را که باید شنود شود، تعیین می‌کند. پروسهٔ صدازنندهٔ این تابع، مادامی که یک پروسهٔ راه دور تقاضای برقراری اتصال با این TSAP را ندهد در حالت توقف باقی خواهد ماند.

دقت کنید که این مدل کاملاً نامتقارن است: یک طرف «غیرفعّال» (Passive) است و با اجرای تابع ***listen*** آنقدر منتظر می‌ماند تا اتفاقی بیفتد [پروسه‌ای ارتباط برقرار کند]. طرف دیگر فعّال (Active) است و در زمان دلخواه شروع به برقراری اتصال می‌کند. یک سؤال جالب آن است که اگر طرف فعّال زودتر شروع کند چه باید کرد. یک استراتژی آن است که اگر هیچ پروسهٔ در حال شنود به TSAP راه دور وجود نداشته نباشد [یعنی سرویس‌دهنده هنوز اجرا نشده باشد]، تلاش برای برقراری ارتباط ناکام گذاشته شود. استراتژی دیگر آن است که پروسهٔ آغازکننده آنقدر منتظر نگاه داشته شود تا بالاخره پروسه‌ای شروع به گوش دادن به TSAP مورد نظر کند.

---

۱. یعنی مثلاً یک بسته در لایهٔ شبکه ممکن است به صورت زنجیره‌ای دهها بار در مسیریابهای واقع بر مسیر پردازش شوند ولی یک بستهٔ TPDU در مبداء تولید و در مقصد پردازش و مصرف می‌شود لذا تمام رخدادهای اتفاقی در لایه‌های زیرین در مقصد قابل بررسی و اصلاح است. مثلاً اگر یک مسیریاب به ناگاه خراب شود و چند بستهٔ TPDU را با خود از بین ببرد هیچ اتفاقی برای لایهٔ انتقال نمی‌افتد چراکه فرستنده متوجه عدم اعلام وصول آنها شده و پس از انقضای مهلت آنها را از نو ارسال می‌کند. ـم

در مثالمان از یک حالت میانه استفاده کرده‌ایم: تقاضای برقراری اتصال برای یک مدت زمان مشخص منتظر نگه داشته می‌شود تا اگر قبل از انقضای مهلت، پروسه‌ای در سمت مقابل تابع ***listen*** را فراخوانی کرد، اتصال برقرار شود. در غیر این صورت تقاضای برقراری اتصال ردّ شده و پروسه صدازننده از حالت توقف خارج و پیغام خطایی به او برگردانده می‌شود.

برای خاتمه دادن به یک اتصال، از تابع کتابخانه‌ای ***disconnect*** بهره گرفته‌ایم. وقتی هر دو طرف ، ارتباط را قطع کردند، اتصال خاتمه می‌یابد. به عبارت دیگر از مدل «متقارن» برای ختم ارتباط استفاده نموده‌ایم.

مسئلهٔ انتقال داده دقیقاً شبیه به برقراری اتصال است: عمل ارسال ماهیّتی «فعال» و فرآیند دریافت ماهیّتی «غیرفعال» دارد؛ لذا برای انتقال داده از راه حل مشابه با برقراری اتصال استفاده خواهیم کرد: فراخوانی تابع ***send*** داده‌ها را فوراً ارسال می‌کند در حالی که فراخوانی تابع ***receive*** غیرفعال بوده و تازمانی که یک بستهٔ TPDU دریافت نشود منجر به توقف پروسه می‌شود.

بدین ترتیب مجموعهٔ سرویسهای ارائه شدهٔ ما، شامل پنج عملکرد اولیه (تابع پایه) است: (۱) CONNECT (۲) LISTEN (۳) DISCONNECT (۴) SEND (۵) RECEIVE. به ازای هر یک از این پنج عملکرد اولیه دقیقاً یک تابع کتابخانه‌ای خاصّ وجود دارد که سرویس مربوطه را پیاده کرده است. پارامترهای این توابع کتابخانه‌ای به صورت زیر هستند:

```
connum = LISTEN(local)
connum = CONNECT(local,remote)
status = SEND(connum,buffer,bytes)
status = RECEIVE(connum,buffer,bytes)
status = DISCONNECT(connnum)
```

تابع اولیهٔ LISTEN اعلام می‌کند که پروسهٔ صدازنندهٔ آن علاقمند به پذیرش آن دسته از تقاضاهای برقراری اتصال است که با شناسهٔ TSAP مشخص شده، وارد می‌شوند. پروسه‌ای که این تابع را فراخوانی کرده، مادامیکه یک پروسهٔ راه دور سعی در برقراری اتصال نکند، متوقف و منتظر خواهد ماند. در اینجا هیچ مهلتی تعیین نشده است.

تابع اولیهٔ CONNECT دو پارامتر را دریافت می‌کند: (۱) شناسهٔ TSAP محلی (Local TSAP) (که در تعریف تابع با نام local ظاهر شده است.) (۲) شناسهٔ TSAP راه دور (با نام remote)؛ سپس سعی می‌کند یک اتصال بین این دو برقرار نماید. اگر این کار با موفقیت انجام شد یک عدد نامنفی در متغیر connum برمی‌گرداند تا برای مشخص کردن هویت اتصال در فراخوانیهای بعدی مورد استفاده قرار بگیرد. اگر ایجاد اتصال با شکست مواجه شود، عددی منفی در connum برمی‌گرداند. در مدل سادهٔ ما، هر TSAP می‌تواند فقط برای ایجاد یک اتصال مورد استفاده قرار بگیرد، لذا یک دلیل موجه برای شکست در برقراری اتصال آن است که آدرس انتقال (یعنی TSAP مورد نظر) در حال استفاده توسط پروسهٔ دیگر باشد. برخی از دلایل دیگر عبارتند از: خاموش بودن ماشین راه دور، آدرس محلی نامعتبر، آدرس راه دور نامعتبر.

تابع اولیهٔ SEND محتویات بافر را به عنوان یک پیام، بر روی اتصال مشخص شده ارسال می‌کند؛ البته در صورت نیاز داده‌ها را در چند قطعه (چند بستهٔ TPDU) می‌فرستد. خطاهای احتمالی در متغیر status برگردانده می‌شود. برخی از علل خطاهای احتمالی عبارتند از: عدم وجود اتصال، نامعتبر بودن آدرس بافر یا منفی بودن پارامتر تعداد بایتی که باید ارسال شود (یعنی منفی بودن پارامتر bytes).

تابع اولیهٔ RECEIVE ، مشخص‌کنندهٔ آن است که پروسهٔ صدازنندهٔ آن تمایل به دریافت داده دارد. اندازهٔ پیام دریافتی در پارامتر bytes قرار داده می‌شود. اگر پروسهٔ راه دور اتصال را قطع کرده باشد یا آدرس بافر نامعتبر (مثلاً

آدرسی خارج از فضای برنامهٔ کاربر) باشد، یک کد خطا در متغیر status برگردانده می‌شود تا ماهیت خطا را مشخص نماید.

تابع اولیه DISCONNECT اتصال مورد نظر را قطع می‌نماید. پارامتر connum ، مشخص می‌کند که کدام اتصال باید بسته شود. علل خطاهای احتمالی عبارتند از: connum شمارهٔ اتصالی متعلق به پروسه‌ای دیگر باشد یا شناسهٔ اتصال نامعتبر باشد. بهر حال در متغیر status ، کدی برگردانده می‌شود که مقدار صفر به معنای موفقیت و مقدار غیرصفر به معنای کد مشخصهٔ خطاست.

۶-۳-۲ واحد انتقال در مثال فوق

قبل از آن که کدهای برنامهٔ «واحد انتقال» (Transport Entity) را بررسی نماییم لطفاً به خاطر بسپارید که این مثال شبیه به مثالهایی که در فصل سوم عرضه شدند جنبهٔ آموزشی دارد و طرحی جدّی محسوب نمی‌شود. برای پرهیز از پیچیدگی، بسیاری از جزئیات فنی (مثل کنترل دقیق و گستردهٔ خطا) که در سیستمهای واقعی مورد نیاز هستند، حذف شده‌اند.

لایهٔ انتقال برای ارسال یا دریافت هر TPDU از سرویسهایی که لایهٔ شبکه در قالب توابع اولیه در اختیار گذاشته، بهره می‌گیرد. در این مثال ابتدا بایستی نوع سرویس لایهٔ شبکه و توابع اولیه مورد استفاده مشخص شود. یک گزینهٔ آن است که از سرویس نامطمئن دیتاگرام بهره گرفته شود. برای آن که مثالمان ساده باشد ما این گزینه را انتخاب نمی‌کنیم. در سرویس دیتاگرام کد برنامهٔ لایهٔ انتقال بسیار طولانی و پیچیده خواهد شد، زیرا بخش بزرگی از این برنامه صرف مدیریت بسته‌های از بین رفته و مسائل ناشی از تأخیر خواهد شد. مضاف بر این، بسیاری از ایده‌های مرتبط با این موضوعات قبلاً به تفصیل در فصل سوم بررسی شده‌اند.

در عوض، ترجیح داده‌ایم از سرویسهای مطمئن یک شبکهٔ اتصال‌گرا بهره بگیریم. بدین ترتیب قادر خواهیم بود که بر روی موضوعاتی از لایهٔ انتقال متمرکز شویم که در لایه‌های زیرین مطرح نمی‌شوند. این موضوعات عبارتند از: برقراری اتصال، خاتمه دادن به اتصال، مدیریت اعتبار (Credit Management). سرویسهای یک لایهٔ انتقال ساده که بر روی شبکهٔ ATM به کار گرفته می‌شود، شبیه به مثال ما خواهد بود.

«واحد انتقال» عموماً بخشی از سیستم عامل ماشین میزبان است؛ گاهی هم در قالب یک مجموعه از توابع کتابخانه‌ای ارائه و در فضای آدرس برنامهٔ کاربر اجرا می‌شود. برای سادگی، برنامهٔ مثال ما در قالب یک مجموعه از توابع کتابخانه‌ای عرضه شده است ولی با تغییرات اندک می‌توان آن را تبدیل به بخشی از سیستم عامل کرد.

به هر حال اشاره به این نکته خالی از لطف نیست که «واحد انتقال» در این مثال یک بخش کاملاً مستقل نیست و بخشی از پروسهٔ کاربردی محسوب می‌شود. خصوصاً وقتی کاربر یک تابع اولیه نظیر LISTEN (که برنامه را بلوکه می‌کند) اجرا نماید، کل واحد انتقال نیز متوقف می‌شود. اینگونه طراحی برای ماشینهایی مناسب است که تک کاربره و تک پروسه‌ای هستند، در حالی که در ماشینی با چندین کاربر و پروسه، طبعاً نیاز به یک واحد انتقال مستقل داریم تا از پروسه‌های کاربران مجزا باشد.

تعامل با لایهٔ شبکه از طریق پروسیجرهای to_net و from_net انجام می‌شود. هر یک از این پروسیجرها، شش پارامتر دارند: اولین پارامتر شناسهٔ اتصال مدار مجازی است. پارامترهای بعدی، بیتهای Q و M هستند که هر گاه به ۱ تنظیم شده باشند، به ترتیب مشخص می‌کنند که (۱) پیام از نوع کنترلی است (۲) بخشی از داده‌های این پیام در بسته‌های بعدی می‌آید. پس از آن نوع بسته مشخص می‌شود که می‌تواند یکی از شش نوع معرفی شده در جدول ۶-۱۹ باشد. سپس اشاره‌گری که آدرس محل شروع داده‌ها را مشخص می‌کند و نهایتاً یک عدد صحیح می‌آید که تعداد بایتهای داده را تعیین می‌نماید.

وقتی پروسیجر to_net فراخوانی می‌شود، «واحد انتقال» تمام این پارامترها را پس از مقداردهی در اختیار لایهٔ

نام بسته لایه شبکه	توصیف
CALL REQUEST	به منظور ایجاد اتصال ارسال می‌شود.
CALL ACCEPTED	پاسخ بسته CALL REQUEST (بمعنای موافقت با برقراری اتصال).
CLEAR REQUEST	به منظور قطع اتصال ارسال می‌شود.
CLEAR CONFIRMATION	پاسخ بسته CLEAR REQUEST (بمعنای موافقت با ختم اتصال).
DATA	برای انتقال داده بکار می‌رود.
CREDIT	یک بسته کنترلی برای مدیریت پنجره

شکل ۶-۱۹. بسته‌هائی از لایهٔ شبکه که در مثالمان از آنها بهره گرفته‌ایم.

شبکه قرار می‌دهد؛ برعکس، با فراخوانی from_net ، لایهٔ شبکه این پارامترها را از بطن بسته‌های ورودی بیرون کشیده و تحویل واحد انتقال می‌دهد. وقتی اطلاعات لازم به صورت پارامترهای پروسیجر به لایهٔ شبکه تحویل می‌شود (به جای آن که این اطلاعات با تحویل بسته‌های واقعی رد و بدل گردد)، لایهٔ انتقال از درگیری با جزئیات پروتکل لایهٔ شبکه دور نگاه داشته می‌شود.[1] هرگاه **«پنجرهٔ لغزان مدار مجازی»**[2] پر شده باشد و واحد انتقال سعی در ارسال بسته‌ای کند ، اجرای to_net آنرا آنقدر معطل نگاه خواهد داشت تا فضای کافی در پنجره آزاد گردد. این مکانیزم با استفاده از دستوراتی شبیه به enanble_transport_layer یا disable_transprot_layer (که در فصل سوم مشابهشان را بررسی کردیم) در لایهٔ شبکه انجام می‌گیرد و بطور کامل از دید واحد انتقال پنهان خواهد ماند. مدیریت پنجره‌ها نیز در لایهٔ شبکه انجام می‌شود.

اضافه بر مکانیزم **«تعلیق»** (suspension) فوق‌الذکر [که توسط لایهٔ شبکه به لایهٔ انتقال تحمیل می‌شود و لایهٔ انتقال نقشی در آن ندارد]، در لایهٔ انتقال دو پروسیجر ***sleep*** و ***wakeup*** (که در فهرست نیامده‌اند) تعریف شده که به منظور ایجاد وقفه در واحد انتقال، مورد استفاده قرار می‌گیرند. پروسیجر ***sleep*** زمانی فراخوانی می‌شود که واحد انتقال منطقاً در انتظار وقوع یک رخداد (مثلاً دریافت یک بسته) باشد. پس از فراخوانی ***sleep***، اجرای واحد انتقال (و به تَبَع آن پروسهٔ کاربردی) متوقف می‌شود.

کُد برنامهٔ «واحد انتقال» در شکل ۶-۲۰ نشان داده شده است. هر اتصال همیشه در یکی از هفت وضعیت زیر قرار دارد:

۱. **وضعیت** IDLE : هنوز هیچ اتصالی برقرار نشده است.

۲. **وضعیت** WAITING : تابع CONNECT اجرا و تقاضای برقراری اتصال (CALL REQUEST) ارسال شده است.

۳. **وضعیت** QUEUED : تقاضای CALL REQUEST دریافت شده ولیکن هنوز LISTEN نشده است.

۴. **وضعیت** ESTABLISHED : اتصال برقرار شده است.

۵. **وضعیت** SENDING : کاربر منتظر دریافت مجوز ارسال یک بسته است.

۶. **وضعیت** RECEIVING : تابع RECEIVE اجرا و در حال دریافت بسته است.

۷. **وضعیت** DISCONNCTING : تابع DISCONNECT جهت قطع اتصال بصورت محلّی اجرا شده است.

---

۱. یعنی بجای آنکه واحد انتقال مستقیماً خودش بسته‌هایی را که قرار است بر روی شبکه ارسال شوند، تولید کند پارامترهای لازم را به لایهٔ شبکه تحویل می‌دهد تا اینکار در لایهٔ شبک انجام شود. -م

۲. Virtual Circuit's Sliding Window

```
#define MAX_CONN 32                  /* max number of simultaneous connections */
#define MAX_MSG_SIZE 8192            /* largest message in bytes */
#define MAX_PKT_SIZE 512             /* largest packet in bytes */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN} cstate;

/* Global variables. */
transport_address listen_address;      /* local address being listened to */
int listen_conn;                       /* connection identifier for listen */
unsigned char data[MAX_PKT_SIZE];      /* scratch area for packet data */

struct conn {
  transport_address local_address, remote_address;
  cstate state;                        /* state of this connection */
  unsigned char *user_buf_addr;        /* pointer to receive buffer */
  int byte_count;                      /* send/receive count */
  int clr_req_received;                /* set when CLEAR_REQ packet received */
  int timer;                           /* used to time out CALL_REQ packets */
  int credits;                         /* number of messages that may be sent */
} conn[MAX_CONN + 1];                  /* slot 0 is not used */

void sleep(void);                      /* prototypes */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{ /* User wants to listen for a connection. See if CALL_REQ has already arrived. */
  int i, found = 0;

  for (i = 1; i <= MAX_CONN; i++)      /* search the table for CALL_REQ */
      if (conn[i].state == QUEUED && conn[i].local_address == t) {
            found = i;
            break;
      }

  if (found == 0) {
      /* No CALL_REQ is waiting.  Go to sleep until arrival or timeout. */
      listen_address = t;  sleep();  i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;         /* connection is ESTABLISHED */
  conn[i].timer = 0;                   /* timer is not used */
```

```
    listen_conn = 0;                         /* 0 is assumed to be an invalid address */
    to_net(i, 0, 0, CALL_ACC, data, 0);      /* tell net to accept connection */
    return(i);                               /* return connection identifier */
}

int connect(transport_address l, transport_address r)
{ /* User wants to connect to a remote process;  send CALL_REQ packet. */
  int i;
  struct conn *cptr;

  data[0] = r;   data[1] = l;                /* CALL_REQ packet needs these */
  i = MAX_CONN;                              /* search table backward */
  while (conn[i].state != IDLE && i > 1) i = i - 1;
  if (conn[i].state == IDLE) {
      /* Make a table entry that CALL_REQ has been sent. */
      cptr = &conn[i];
      cptr->local_address = l; cptr->remote_address = r;
      cptr->state = WAITING; cptr->clr_req_received = 0;
      cptr->credits = 0; cptr->timer = 0;
      to_net(i, 0, 0, CALL_REQ, data, 2);
      sleep();                               /* wait for CALL_ACC or CLEAR_REQ */
      if (cptr->state == ESTABLISHED) return(i);
      if (cptr->clr_req_received) {
            /* Other side refused call. */
            cptr->state = IDLE;              /* back to IDLE state */
            to_net(i, 0, 0, CLEAR_CONF, data, 0);
            return(ERR_REJECT);
      }
  } else return(ERR_FULL);                   /* reject CONNECT: no table space */
}

int send(int cid, unsigned char bufptr[], int bytes)
{ /* User wants to send a message. */
  int i, count, m;
  struct conn *cptr = &conn[cid];

  /* Enter SENDING state. */
  cptr->state = SENDING;
  cptr->byte_count = 0;                      /* # bytes sent so far this message */
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
      /* Credit available; split message into packets if need be. */
      do {
            if (bytes - cptr->byte_count > MAX_PKT_SIZE) {/* multipacket message */
                  count = MAX_PKT_SIZE;  m = 1;/* more packets later */
            } else {                         /* single packet message */
                  count = bytes - cptr->byte_count;  m = 0;/* last pkt of this message */
            }
            for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
            to_net(cid, 0, m, DATA_PKT, data, count);/* send 1 packet */
            cptr->byte_count = cptr->byte_count + count;/* increment bytes sent so far */
      } while (cptr->byte_count < bytes);  /* loop until whole message sent */
```

```
cptr->credits--;                    /* each message uses up one credit */
      cptr->state = ESTABLISHED;
      return(OK);
 } else {
      cptr->state = ESTABLISHED;
      return(ERR_CLOSED);           /* send failed: peer wants to disconnect */
 }
}

int receive(int cid, unsigned char bufptr[], int *bytes)
{ /* User is prepared to receive a message. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
      /* Connection still established; try to receive. */
      cptr->state = RECEIVING;
      cptr->user_buf_addr = bufptr;
      cptr->byte_count = 0;
      data[0] = CRED;
      data[1] = 1;
      to_net(cid, 1, 0, CREDIT, data, 2); /* send credit */
      sleep();                            /* block awaiting data */
      *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
{ /* User wants to release a connection. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {       /* other side initiated termination */
      cptr->state = IDLE;             /* connection is now released */
      to_net(cid, 0, 0, CLEAR_CONF, data, 0);
  } else {                            /* we initiated termination */
      cptr->state = DISCONN;          /* not released until other side agrees */
      to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}

void packet_arrival(void)
{ /* A packet has arrived, get and process it. */
  int cid;                            /* connection on which packet arrived */
  int count, i, q, m;
  pkt_type ptype;    /* CALL_REQ, CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT, CREDIT */
  unsigned char data[MAX_PKT_SIZE]; /* data portion of the incoming packet */
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count); /* go get it */
  cptr = &conn[cid];
```

```
switch (ptype) {
  case CALL_REQ:                    /* remote user wants to establish connection */
    cptr->local_address = data[0]; cptr->remote_address = data[1];
    if (cptr->local_address == listen_address) {
         listen_conn = cid; cptr->state = ESTABLISHED; wakeup();
    } else {
         cptr->state = QUEUED; cptr->timer = TIMEOUT;
    }
    cptr->clr_req_received = 0; cptr->credits = 0;
    break;
  case CALL_ACC:                    /* remote user has accepted our CALL_REQ */
    cptr->state = ESTABLISHED;
    wakeup();
    break;
  case CLEAR_REQ:                   /* remote user wants to disconnect or reject call */
    cptr->clr_req_received = 1;
    if (cptr->state == DISCONN) cptr->state = IDLE;/* clear collision */
    if (cptr->state == WAITING || cptr->state == RECEIVING || cptr->state == SENDING) wakeup();
    break;
  case CLEAR_CONF:                  /* remote user agrees to disconnect */
    cptr->state = IDLE;
    break;
  case CREDIT:                      /* remote user is waiting for data */
    cptr->credits += data[1];
    if (cptr->state == SENDING) wakeup();
    break;
  case DATA_PKT:                    /* remote user has sent data */
    for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] = data[i];
    cptr->byte_count += count;
    if (m == 0 ) wakeup();
  }
}

void clock(void)
{ /* The clock has ticked, check for timeouts of queued connect requests. */
 int i;
 struct conn *cptr;

 for (i = 1; i <= MAX_CONN; i++) {
     cptr = &conn[i];
     if (cptr->timer > 0) {         /* timer was running */
         cptr->timer--;
         if (cptr->timer == 0) {    /* timer has now expired */
             cptr->state = IDLE;
             to_net(i, 0, 0, CLEAR_REQ, data, 0);
         }
     }
 }
}
```

شکل ۶-۲۰. مثالی از کُد برنامهٔ واحد انتقال (Transport Entity).

گذار از یک وضعیت به وضعیتی دیگر زمانی اتفاق می‌افتد که یکی از وقایع ذیل اتفاق بیفتد: یک تابع اولیه اجرا شود، بسته‌ای دریافت شود یا مهلت تایمر منقضی شود.

پروسیجرهای نشان داده شده در شکل ۶-۲۰ بر دو نوعند: اغلب آنها مستقیماً قابل فراخوانی در برنامهٔ کاربر هستند در حالیکه ***Packet-arrival*** و ***clock*** متفاوتند. این دو تابع در اثر رخدادهای خارجی به صورت خودکار شروع به کار می‌کنند: اولی در اثر دریافت یک بسته و دومی با هر تیک ساعت به کار می‌افتد. در حقیقت این دو، روتینهای وقفه (Interrupt Routines) هستند. فرض خواهیم کرد که این دو تابع در حین اجرای هیچیک از پروسیجرهای واحد انتقال فراخوانی نمی‌شوند بلکه فقط زمانی که پروسهٔ کاربر در حالت توقّف (sleeping) یا در حال اجرای کُدی غیر از پروسیجرهای لایهٔ انتقال باشد، فراخوانی می‌شوند. این ویژگی برای عملکرد صحیح کُد برنامه، الزامی و حیاتی است.

وجود بیت Q (Qualifer) در سرآیند بسته اجازه می‌دهد که از سربار پروتکل لایهٔ انتقال جلوگیری شود: پیامهای حاوی داده‌های معمولی، به صورت بسته‌هایی با Q=0 ارسال می‌شوند؛ پیامهای کنترلی لایهٔ انتقال (که در مثال ما فقط یک پیام و آن هم CREDIT است) در قالب بسته‌ای ارسال می‌شوند که در آنها Q=1 است. این پیامهای کنترلی در سمت گیرنده توسط «واحد انتقال» کشف و پردازش می‌شوند.

ساختمان دادهٔ اصلی که توسط واحد انتقال مورد استفاده قرار می‌گیرد، آرایهٔ conn است که به ازای هر اتصال یک رکورد در آن ذخیره می‌شود. این رکورد وضعیت فعلی یک اتصال را نگه می‌دارد و شامل اطلاعات ذیل است: (۱) آدرس انتقال طرف مقابل (یعنی آدرس TSAP طرف مقابل) (۲) تعداد پیامهای ارسالی یا دریافتی از آن اتصال (۳) وضعیت جاری [یکی از هفت وضعیت] (۴) اشاره‌گر آدرس بافر کاربر (۵) تعداد بایتهایی که تاکنون از پیام فعلی ارسال یا دریافت شده‌اند. (۶) یک بیت که مشخص می‌کند کاربر راه دور، دستور DISCONNECT را صادر کرده است. (۷) یک «شمارندهٔ مجوز» (Permission Counter) که ارسال پیامها را فعال و ممکن می‌سازد. البته در مثال سادهٔ ما از تمام این فیلدها استفاده نمی‌شود بلکه یک «واحد انتقال» کامل و دقیق به این فیلدها و حتی فیلدهای بیشتر، نیاز خواهد داشت. فرض بر آنست که هر درایه از آرایهٔ conn در وضعیت اولیهٔ IDLE قرار می‌گیرد. [یعنی با IDLE مقداردهی اولیه می‌شود.]

وقتی کاربری تابع CONNECT را فراخوانی می‌کند، به لایهٔ شبکه دستور داده می‌شود که یک بسته از نوع CALL REQUEST برای ماشین راه دور بفرستد؛ سپس کاربر به حالت توقف (استراحت) می‌رود. وقتی بستهٔ CALL REQUEST در طرف مقابل دریافت شود، به «واحد انتقال» وقفه (اینتراپت) داده می‌شود تا با اجرای روتین packet_arrival بررسی کند که آیا یک کاربر محلی به آدرس مشخص شده گوش می‌دهد یا خیر؛ اگر اینگونه باشد، بستهٔ CALL ACCEPTED بازگردانده شده و کاربر راه دور مجدداً فعال می‌شود. اگر اینگونه نباشد، CALL REQUEST در صف انتظار قرار می‌گیرد تا ساعت، به تعداد TIMEOUT تیک بزند. اگر در خلال این مدت عمل LISTEN انجام شود، اتصال برقرار خواهد شد؛ در غیر این صورت مهلت زمان، منقضی و با ارسال بستهٔ CLEAR REQUEST این اتصال ردّ خواهد شد تا طرف مقابل تا ابد در حالت توقف نماند.

اگرچه در مثالمان سرآیند پروتکل انتقال را حذف کرده‌ایم ولی بهرحال به راهی نیاز داریم تا بتوان متوجه شد که هر بسته متعلق به کدام اتصال است، چراکه ممکن است بطور همزمان چندین اتصال فعّال وجود داشته باشد. ساده‌ترین راه حل آن است که از شمارهٔ مدار مجازی در لایهٔ شبکه به عنوان شمارهٔ اتصال در لایه انتقال استفاده شود. مضاف بر این، از شمارهٔ مدار مجازی می‌توان به عنوان اندیس آرایهٔ conn بهره گرفت. یعنی وقتی بسته‌ای به مدار مجازی شمارهٔ k در لایهٔ شبکه وارد شود متعلق به اتصال شمارهٔ k در لایهٔ انتقال می‌باشد، و وضعیت این اتصال در رکورد conn[k] قرار گرفته است. برای اتصالاتی که از یک ماشین میزبان شروع می‌شود، شمارهٔ اتصال

توسط «واحد انتقال مبداء»[1] انتخاب می‌شود. برای تقاضاهای اتصال، لایهٔ شبکه این انتخاب را انجام می‌دهد و طبعاً یک شمارهٔ بلااستفاده از بین شماره‌های مدار مجازی انتخاب می‌کند.

برای آن که مجبور به مدیریت بافرها در درون واحد انتقال نباشیم از یک مکانیزم کنترل جریان خاصّ و متفاوت از روش معمولی پنجرهٔ لغزان استفاده کرده‌ایم. وقتی یک کاربر، تابع RECEIVE را فراخوانی می‌کند یک پیام خاص به نام CREDIT (پیام اعتبار) برای واحد انتقال در ماشین فرستنده ارسال می‌شود و در آرایهٔ conn ثبت می‌شود. وقتی تابع SEND فراخوانی شود، واحد انتقال ابتدا بررسی می‌کند که آیا برای اتصال مربوطه **«اعتبار»** لازم وجود دارد یا خیر: اگر اعتبار لازم وجود داشته باشد، پیام ارسال می‌شود (در صورت لزوم در چند بسته)؛ سپس به اندازهٔ طول پیام از اعتبار موجود کاسته می‌شود؛ در غیر این صورت، واحد انتقال خودش را به حالت توقف (Sleep) می‌برد تا اعتبار لازم برسد. این مکانیزم تضمین می‌کند که هیچ پیامی ارسال نشود مگر آن که طرف مقابل قبلاً فرمان RECEIVE را اجرا کرده باشد. لذا وقتی پیامی از راه برسد مطمئناً بافر لازم برای آن موجود و آماده خواهد بود. این روش را می‌توان براحتی تعمیم داد تا گیرندگان بتوانند چندین بافر تهیه کرده و تقاضای چند پیام بدهند.

سادگی کُد برنامهٔ نشان داده شده در شکل ۶-۲۰ را به خاطر بسپارید. یک واحد انتقال واقعی باید اعتبار تمام پارامترهای عرضه شده توسط کاربر را بررسی نماید، جبران از کار افتادگی در لایهٔ شبکه را بر عهده بگیرد، برخورد دو تماس (Call Collisions) را مدیریت کند و از سرویسهای انتقال عمومی‌تری شامل تسهیلات وقفه، سرویس دیتاگرام و «نسخه‌های غیرقابل تعلیق SEND و RECEIVE»[2] پشتیبانی نماید.

۶-۳-۳ بررسی مثال فوق از دید «ماشین حالت محدود»

نوشتن یک برنامه برای «واحد انتقال» خصوصاً برای پروتکلهای واقعی و عملی‌تر، بسیار دشوار است. برای آن که احتمال خطا کاهش یابد، توصیف وضعیت پروتکل در قالب یک **«ماشین حالت محدود»** (Finite State Machine) اغلب سودمند است. قبلاً دیدیم که در پروتکل مثال ما، برای هر اتصال هفت وضعیت وجود دارد. در مجموع می‌توان ۱۲ رخداد مختلف تعریف کرد که می‌توانند وضعیت یک اتصال را تغییر بدهند. پنج تا از این رخدادها، در اثر فراخوانی توابع اولیه حادث می‌شوند. شش‌تای بعدی در اثر ورود ۶ بستهٔ معتبر رخ می‌دهند. آخرین رخداد ناشی از انقضای مهلت تایمر است. در شکل ۶-۲۱ فعالیتهای اصلی پروتکل به شکل ماتریسی نشان داده شده است. ستونها بیانگر «وضعیت» و سطرها بیانگر ۱۲ «رخداد» مختلف است.

هر درایه (Entry) در ماتریس شکل ۶-۲۱ (یا به عبارتی در ماشین حالت محدود) حداکثر سه فیلد دارد: (۱) **گزاره (Predicate) (۲) کُنش (Action) (۳) وضعیت جدید (new state)**

«گزاره» مشخص می‌کند که تحت چه شرایطی «کُنش» تعیین شده انجام خواهد گرفت. به عنوان نمونه، درایهٔ گوشهٔ بالا و سمت چپ ماتریس نشان می‌دهد که اگر LISTEN اجرا شود ولی در جدول، فضای حافظه باقی نمانده باشد (گزارهٔ P1)، اجرای LISTEN با شکست مواجه شده و وضعیت پروتکل تغییر نخواهد کرد. از طرف دیگر، اگر یک بستهٔ درخواست CALL REQUEST برای آدرسی که یک پروسه در حال گوش دادن به آن است، بیاید (گزارهٔ P2)، اتصال مربوطه بلافاصله برقرار می‌شود. حالت ممکن بعدی آن است که گزارهٔ P2 نادرست باشد یعنی هیچ بستهٔ CALL REQUEST دریافت نشده باشد که در این حالت، اتصال در وضعیت بیکار (IDLE) و در انتظار بستهٔ CALL REQUEST باقی خواهد ماند.

اشاره به این نکته مهم است که انتخاب وضعیتهایی که در ماتریس فوق بکار رفته است، در برگیرندهٔ تمام

---

۱. Originating Transport Entity
۲. nonblocking SEND/RECEIVE

		State						
		Idle	Waiting	Queued	Established	Sending	Receiving	Dis-connecting
Primitives (فراخوانی عملکردهای اولیه)	LISTEN	P1: ~/Idle P2: A1/Estab $\overline{P2}$: A2/Idle		~/Estab				
	CONNECT	P1: ~/Idle $\overline{P1}$: A3/Wait						
	DISCONNECT				P4: A5/Idle $\overline{P4}$: A6/Disc			
	SEND				P5: A7/Estab $\overline{P5}$: A8/Send			
	RECEIVE				A9/Receiving			
Incoming packets (بسته‌های ورودی)	Call_req	P3: A1/Estab $\overline{P3}$: A4/Queu'd						
	Call_acc		~/Estab					
	Clear_req		~/Idle		A10/Estab	A10/Estab	A10/Estab	~/Idle
	Clear_conf							~/Idle
	DataPkt						A12/Estab	
	Credit				A11/Estab	A7/Estab		
Clock	Timeout			~/Idle				

Predicates
P1: Connection table full
P2: Call_req pending
P3: LISTEN pending
P4: Clear_req pending
P5: Credit available

Actions
A1: Send Call_acc
A2: Wait for Call_req
A3: Send Call_req
A4: Start timer
A5: Send Clear_conf
A6: Send Clear_req
A7: Send message
A8: Wait for credit
A9: Send credit
A10: Set Clr_req_received flag
A11: Record credit
A12: Accept message

شکل ۶-۲۱. مدل پروتکل مثال قبل در قالب «ماشین حالت محدود». هر درایه (Entry) یک «گزاره» اختیاری ، یک «کُنش» اختیاری و یک «وضعیت» جدید دارد. علامت ~ بیانگر آنست که هیچ «کنشی» صورت نمی‌گیرد. علامت - بر روی یک گزاره، منفی آنرا نشان می‌دهد. درایه‌های خالی مربوط به رخدادهای ناممکن یا نامعتبر هستند.

وضعیتهای ممکن نیست: در این مثال، وضعیت LISTENING (یعنی وضعیت انتظار برای برقراری اتصال) وجود ندارد در حالی که این وضعیت، پس از صدور فرمان LISTEN اتفاق می‌افتد و وضعیتی معقول به شمار می‌رود. چرا این وضعیت لحاظ نشده است؟ زیرا تصمیم گرفته‌ایم که از شمارهٔ مدار مجازی بکار رفته در لایهٔ شبکه به عنوان شناسهٔ اتصال (Connection Identifier) استفاده کنیم و به واسطهٔ صدور فرمان LISTEN هیچ رکورد اتصال خاصّی که مبیّن وضعیت LISTENING باشد در جدول اتصالات ایجاد نخواهد شد. یعنی برای LISTEN شمارهٔ مدار مجازی، نهایتاً توسط لایهٔ شبکه و در هنگام ورود بستهٔ CALL REQUEST انتخاب می‌شود.

کُنشهای A1 تا A12 ، عملیات اصلی تلقی می‌شوند؛ عملیاتی نظیر ارسال بسته و راه‌اندازی تایمرها. عملیات

کوچک و جانبی مثل مقداردهی اولیه به فیلدهای یک رکورد اتصال، فهرست نشده‌اند. اگر یک «کنش» منوط به فعال کردن یک پروسهٔ معلق باشد، عملیات لازم پس از فعال‌سازی پروسه نیز جزو آن کنش محسوب می‌شود: به عنوان مثال هر گاه بستهٔ CALL REQUEST وارد شود و پروسه‌ای در انتظار آن معلق و منتظر مانده باشد، ارسال بستهٔ CALL ACCEPT که بلافاصله پس از فعال شدن پروسه انجام می‌گیرد به عنوان بخشی از عمل CALL REQUEST محسوب می‌شود. پس از انجام هر عمل، اتصال در وضعیت جدیدی قرار می‌گیرد. (به شکل ۶-۲۱ دقت کنید.)

توصیف عملکرد پروتکل در قالب ماتریس سه مزیت دارد: اول آن که بدین شکل برنامه‌نویس ساده‌تر می‌تواند ترکیبات مختلف «وضعیت» و «رخدادها» را بررسی کرده و نیاز به یک «کنش» را تشخیص بدهد. در پیاده‌سازی عملی و واقعی پروتکل، برخی از این ترکیبات (که اکنون به حال خود رها شده‌اند) برای مدیریت خطا کاربرد دارند. در شکل ۶-۲۱ هیچ تمایزی بین «وضعیتهای غیرممکن» و «وضعیتهای نامعتبر» قائل نشده‌ایم.[۱] به عنوان مثال هرگاه یک اتصال در وضعیت «انتظار» (waiting) قرار داشته باشد، رخداد DISCONNECT غیرممکن است چراکه در وضعیت انتظار، پروسهٔ کاربر متوقف شده و هرگز نمی‌تواند هیچ تابعی را اجرا نماید. برعکس، در وضعیت sending (ارسال)، انتظار دریافت بسته نمی‌رود زیرا هیچ اعتباری جهت ارسال برای طرف مقابل صادر نشده است و دریافت یک بستهٔ داده، خطای پروتکل محسوب می‌شود.

دومین مزیت نمایش ماتریسی پروتکل، پیاده‌سازی آنست. برنامه‌نویس می‌تواند آرایه‌ای دو بعدی در نظر بگیرد که در هر عنصر آن یعنی $a[i][j]$، اشاره‌گر شروع یا اندیس پروسیجری قرار گرفته که هنگام بروز رخداد i در وضعیت j، باید اجرا شود و ادارهٔ امور را بر عهده بگیرد. یکی از راهکارهای پیاده‌سازی «واحد انتقال» آنست که برنامه را در قالب یک حلقهٔ کوتاه (حلقهٔ تکرار در برنامه‌نویسی) بنویسیم که در ابتدای حلقه منتظر وقوع یک رخداد باقی می‌ماند. وقتی یک رخداد اتفاق می‌افتد اتصال مربوطه شناسایی و تعیین شده و وضعیت آن استخراج می‌گردد. براساس وضعیت و رخدادهایی که در این ماتریس تبیین شده، «واحد انتقال» اندیس مناسب را استخراج و با رجوع به آرایهٔ a، پروسیجر مناسب را فراخوانی و اجرا می‌نماید.

حُسن سوم در روش «ماشین حالت محدود»، سادگی توصیف پروتکل است. در مستندات برخی از استانداردها، هر پروتکل در قالب یک «ماشین حالت محدود» (همانند آنچه که در شکل ۶-۲۲ نشان داده شده)، توصیف و تشریح شده است. هر گاه واحد انتقال در قالب یک ماشین حالت محدود توصیف شده باشد راحتتر می‌توان از طریق آن به سوی پیاده‌سازی عملی آن حرکت کرد.

اشکال روش «ماشین حالت محدود» آن است که فهم و تحلیل آن دشوارتر از برنامه‌نویسی معمولی است. البته این اشکال را می‌توان با ترسیم ماشین حالت محدود به صورت گراف، تا حدودی حل کرد. این نوع نمایش در شکل ۶-۲۲ نشان داده شده است.

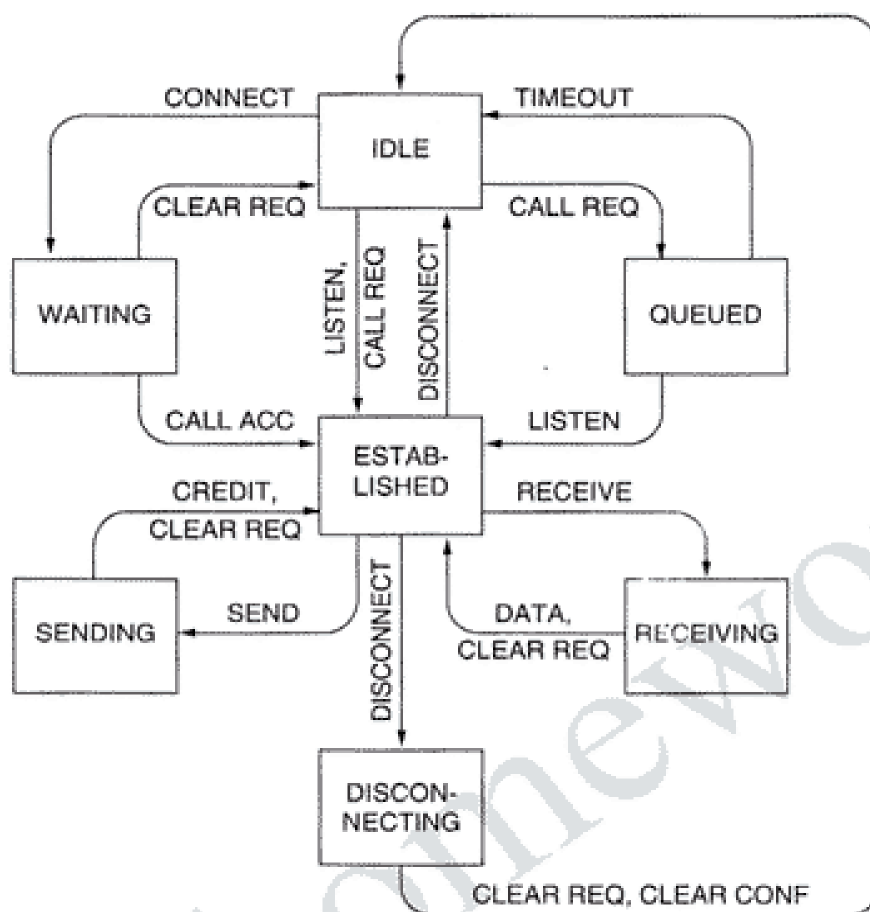
۴-۶ پروتکلهای لایهٔ انتقال در اینترنت: UDP[۲]

اینترنت در لایهٔ انتقال دو پروتکل عمده دارد: یکی بدون‌اتصال و دیگری اتصال‌گرا. در بخشهای آتی به مطالعهٔ این دو پروتکل خواهیم پرداخت. پروتکل بدون‌اتصال در اینترنت، UDP و پروتکل اتصال‌گرای آن، TCP نام دارد. از آنجایی که UDP اساساً همان IP به همراه یک سرآیند کوتاه است لذا با آن شروع خواهیم کرد. در ضمن به دو کاربرد مختلف UDP خواهیم پرداخت.

---

۱. «وضعیتهای غیرممکن» یعنی وضعیتهایی که هرگز اتفاق نخواهد افتاد و «وضعیتهای نامعتبر» وضعیتهایی است که نباید اتفاق بیفتد! -م

۲. User Datagram Protocol



شکل ۶-۲۲. مدل پروتکل مثال قبلی در قالب گراف. برای سادگی «گذارهایی» (Transitions) که وضعیت اتصال را تـغییر نمی‌دهند از گراف حذف شده‌اند.

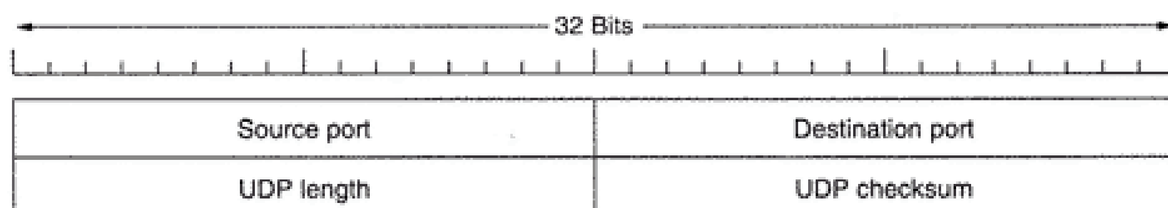
۶-۴-۱ مقدمه‌ای بر UDP

پشتهٔ پروتکلی اینترنت در لایهٔ انتقال از یک پروتکل بدون‌اتصال به نام UDP پشتیبانی می‌کند. UDP امکان آن را فراهم آورده که برنامه‌های کاربردی بتوانند بدون ایجاد هر گونه اتـصال و هـماهنگی قـبلی، داده‌ای را درون یک دیتاگرام IP جاسازی کرده و آن را بفرستند. پروتکل UDP در RFC 768 تشریح شده است.

UDP داده‌ها را در قالب قطعاتی[1] ارسال می‌کند که در ابتدای آنها ۸ بایت سرآیند و سپس داده‌های لایهٔ کاربرد قرار می‌گیرد. این سرآیند در شکل ۶-۲۳ نشان داده شده است. دو فیلد شمارهٔ پورت[2] به منظور شناسائی نقاط پایانی (پروسه‌های نهایی) در ماشینهای مبداء و مقصد به کار می‌آیند. وقتی یک بستهٔ UDP از راه می‌رسد، محتوای آن به پروسهٔ متصل[3] به شمارهٔ پورت مقصد ، تحویل داده می‌شود. عمل اتصال پروسه به یک پورت از طریق تابع اولیهٔ BIND (که تعریف آنرا برای TCP در شکل ۶-۶ دیدیم) انجام می‌شود. (فرآیند مقیدسازی پروسه به یک پورت در TCP یا UDP تفاوتی ندارد.) در حقیقت، آنچه که UDP در مقایسه با IP معمولی اضافه‌تر دارد پورتهای مبداء و مقصد هستند. بدون فیلدهای مربوط به پورت، لایهٔ انتقال نمی‌داند که با یک بسته چه کار کند. با این فیلدها، قطعه داده به درستی تحویل پروسهٔ مربوطه خواهد شد.

---

۱. Segments
۲. Source & Destination Port
۳. Attach



شکل ۶-۲۳. سرآیند UDP ‏(UDP Header).

برای آن که بتوان برای پروسهٔ مبداء پاسخی برگرداند، به شمارهٔ پورت مبداء نیاز است. بدین منظور محتوای فیلد پورت مبداء (Source Port) از بستهٔ ورودی، در فیلد پورت مقصد از بستهٔ خروجی، کپی و ارسال می‌شود. بدین ترتیب فرستندهٔ پاسخ ، پروسهٔ تحویل گیرندهٔ بسته را مشخص می‌نماید.

فیلد UDP Length اندازهٔ کل قطعهٔ داده (شامل ۸ بایت سرآیند) را مشخص می‌نماید. فیلد UDP checksum به منظور کشف خطاهای احتمالی کاربرد دارد و اختیاری است؛ در صورت عدم محاسبه، در این فیلد عدد صفر درج می‌شود. (البته اگر مقدار واقعی این کُد صفر باشد به جای آن تمام فیلد با بیتهای ۱ پر می‌شوند.) عدم استفاده از این فیلد نوعی سهل‌انگاری است مگر آن که کیفیت داده‌ها چندان مهم نباشد. (مثلاً در مورد صدای دیجیتال)

شاید اشارهٔ صریح به کارهایی که UDP انجام نمی‌دهد، ارزشمند باشد. این پروتکل کنترل جریان (Flow Control) و کنترل خطا (Error Control) ندارد و در صورت دریافت یک قطعه دادهٔ خراب، آن را «ارسال مجدّد» نخواهد کرد. تمام این عملیات بر عهدهٔ پروسهٔ کاربر گذاشته شده است. آنچه که این پروتکل انجام می‌دهد عبارت است از ایجاد یک واسط (Interface) بین پروسه‌های کاربردی و پروتکل IP و انجام عمل دی‌مالتی‌پلکس قطعات داده بین پروسه‌های کاربردی.[1] این تمام کاری است که UDP انجام می‌دهد. برای برنامه‌های کاربردی که نیاز به کنترل دقیق و جدی جریان، کنترل خطا و زمان‌بندی دارند، UDP امکانات چندانی را در اختیار نمی‌گذارد و آن را به خود برنامه محول کرده است.

یکی از زمینه‌هایی که استفاده از UDP مفید است، برنامه‌های خاص سرویس‌ده/مشتری هستند که در آنها مشتری تقاضای کوتاهی را برای سرویس‌دهنده می‌فرستد و انتظار پاسخی کوتاه دارد. اگر پیام تقاضا یا پاسخ از بین برود، مهلت مشتری به سر می‌آید و از نو شروع می‌کند. در این حالت، نه تنها کد برنامه ساده‌تر می‌شود بلکه به مبادله پیامهای کمتری در دو جهت نیاز است.

یکی از برنامه‌های کاربردی که از UDP بهره گرفته، DNS[2] است که آن را در فصل هفتم مطالعه خواهیم کرد. خلاصهٔ عملکرد DNS بدین نحو است: یک برنامه که در جستجوی آدرس IP یک ماشین میزبان با نامی مثل www.cs.berkeley.edu است یک بستهٔ UDP حاوی نام ماشین میزبان به سوی DNS می‌فرستد. سرویس‌دهندهٔ DNS، با ارسال یک بستهٔ UDP، آدرس IP ماشین میزبان مورد نظر را به اطلاع ماشین سوال کننده می‌رساند. در این حالت نیازی به هماهنگی قبلی و ایجاد یا ختم اتصال نیست و در مجموع فقط دو پیام ردّ و بدل می‌شود.

۶-۴-۲ فراخوانی پروسیجرهای راه دور (RPC : Remote Procedure Call)

از برخی جهات، ارسال یک پیام برای یک ماشین راه دور و دریافت پاسخ، با فراخوانی یک تابع در زبانهای

---

۱. یعنی با بررسی شمارهٔ پورت مقصد از هر بستهٔ UDP آنرا بین پروسه‌های مربوطه توزیع می‌نماید. م-
۲. Domain Name System

برنامه‌نویسی مشابهت دارد. در هر دو مورد یک یا چند پارامتر را تحویل می‌دهید و نتیجه‌ای به شما بازگردانده می‌شود. این شباهت، افراد را بدین نکته رهنمون ساخت که عملیات مبتنی بر «پرسش و پاسخ[1]» در شبکه» را می‌توان در قالب فراخوانی پروسیجر سازماندهی و تنظیم کرد. چنین ساختاری، برنامه‌نویسی برنامه‌های کاربردی شبکه را ساده‌تر و از لحاظ مفهومی آشناتر و ملموس‌تر می‌کند. به عنوان مثال، پروسیجری به نام get_IP_address(host_name) را مدّ نظر قرار بدهید که با ارسال یک بستهٔ UDP برای سرویس‌دهندهٔ DNS کار خود را آغاز کرده و منتظر پاسخ آن باقی می‌ماند و اگر پاسخ آن در مهلت مقرر دریافت نشود، تلاش خود را تکرار می‌کند. بدین ترتیب تمام جزئیات شبکه از دید برنامه‌نویس مخفی می‌ماند.

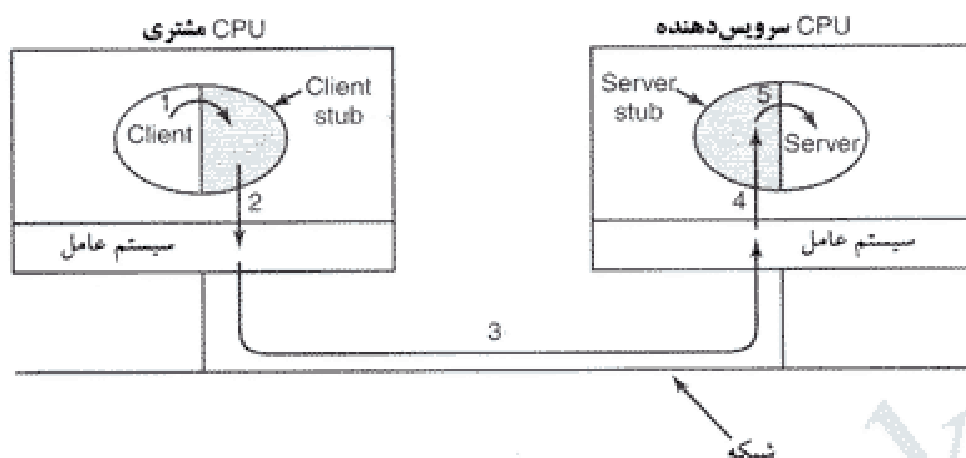
در این زمینه کارهایی اساسی توسط دو نفر به نامهای Birrell و Nelson (۱۹۸۴) انجام شده است. چکیدهٔ آنچه که Birrell و Nelson پیشنهاد کردند آنست که به برنامه‌ها اجازه داده شود پروسیجرهایی را بر روی ماشینهای راه دور فراخوانی نمایند. وقتی پروسه‌ای بر روی ماشین ۱ ، پروسیجری را بر روی ماشین ۲ فراخوانی می‌کند، پروسهٔ صدازننده بر روی ماشین ۱ متوقف و معلق شده و اجرای پروسیجر بر روی ماشین ۲ آغاز می‌شود. اطلاعات لازم از پروسهٔ صدازننده، در قالب چند پارامتر تحویل پروسیجر شده و نتیجهٔ اجرای پروسیجر در پارامترهایی بازگردانده می‌شود. بدین نحو ردّ و بدل هرگونه پیام، از دید برنامه‌نویس مخفی می‌ماند. این تکنیک اصطلاحاً RPC نامیده می‌شود (Remote Procedure Call) و به زیربنای بسیاری از برنامه‌های کاربردی شبکه تبدیل شده است. بطور مرسوم پروسیجر صدازننده با عنوان «مشتری» و پروسیجر فراخوانی شده، با عنوان «سرویس‌دهنده» معرفی می‌شوند و ما نیز از همین اسامی بهره می‌گیریم.

ایده‌ای که در ورای RPC قرار دارد آن است که فراخوانی پروسیجرهای راه دور حتی‌الامکان شبیه به فراخوانی پروسیجرهای محلی باشد. در ساده‌ترین شکل ممکن، برای فراخوانی پروسیجرهای راه دور، باید به برنامهٔ مشتری یک پروسیجر کتابخانه‌ای کوچک به نام client stub مقیّد (bind) شود که پروسیجر سرویس‌دهنده را در فضای آدرس برنامهٔ مشتری تصویر می‌کند. به روش مشابه، سرویس‌دهنده نیز به پروسیجر کوچکی به نام server stub مقید می‌شود که وظیفهٔ آن، پنهان‌سازی این واقعیت است که فراخوانی پروسیجر توسط یک برنامهٔ مشتری بر روی سرویس‌دهندهٔ محلی انجام نشده است.

در شکل ۶-۲۴، مراحل واقعی یک فراخوانی RPC نشان داده شده است. در مرحلهٔ ۱ ، برنامهٔ مشتری پروسیجر client stub را فراخوانی می‌کند. این مرحله، یک فراخوانی پروسیجر محلی است و طبق معمول پارامترهای لازم به درون پشته هدایت می‌شوند. در مرحلهٔ ۲، پروسیجر client stub این پارامترها را در یک پیام جاسازی کرده و برای ارسال آن یک فراخوانی سیستمی انجام می‌دهد. جاسازی پارامترها در یک پیام اصطلاحاً عمل «**مارشالینگ**» (تشریفات) نام دارد. در مرحلهٔ سوم هستهٔ سیستم عامل، این پیام را از ماشین مشتری به سوی ماشین سرویس‌دهنده ارسال می‌کند. در مرحلهٔ چهارم هستهٔ سیستم عامل سرویس‌دهنده، بستهٔ ورودی را به server stub تحویل می‌دهد. نهایتاً در مرحلهٔ پنجم، server stub پروسیجر سرویس‌دهنده را با پارامترهای دریافتی اجرا می‌نماید. فرآیند ارسال پاسخ در جهت مقابل نیز طبق همین روال انجام خواهد شد.

اشاره به این نکتهٔ کلیدی مهم است که در برنامهٔ مشتری که کاربر آن را می‌نویسد، فراخوانی پروسیجر client stub به روش معمولی انجام می‌شود و نام آن با نام پروسیجر سرویس‌دهنده یکی است. از آنجایی که برنامهٔ مشتری و پروسیجر client stub در فضای آدرس مشابهی قرار دارند لذا تحویل پارامترهای لازم برای فراخوانی پروسیجر به روش معمول (و از طریق پشته) انجام می‌گیرد. به روش مشابه، پروسیجر سرویس‌دهنده

۱. request & reply



شکل ۶-۲۴. مراحل لازم برای عمل فراخوانی پروسیجر راه دور (RPC). Client Stub و Server Stub نشان داده نشده است.

توسط پروسیجر دیگر یعنی server stub (در فضای آدرس خودش و با پارامترهای لازم) به روش معمولی فراخوانی می‌شود. برای پروسیجرهای سرویس‌دهنده هیچ چیزی غیرمعمول و نامتعارف نیست. بدین ترتیب به جای آن که عملیات I/O از طریق سوکت انجام شود، مبادلهٔ اطلاعات در شبکه با شبیه‌سازی و تقلید فراخوانی پروسیجر انجام می‌گیرد.

فارغ از زیبایی مفهوم RPC، مار در آستین آن نهان است!! مهمترین مشکل آن پارامترهای نوع اشاره‌گر هستند. بطور طبیعی، ارسال یک اشاره‌گر به پروسیجرهای محلی، اشکالی ندارد. پروسیجر محلی می‌تواند از اشاره‌گرها به همان نحوی که پروسهٔ صدازننده از آنها بهره می‌گیرد، استفاده کند چرا که هر دوی آنها در فضای آدرس مجازی یکسانی به سر می‌برند. در RPC ، ارسال پارامترهایی که از نوع اشاره‌گر هستند غیرممکن است چرا که مشتری و سرویس‌دهنده در فضای آدرس متفاوتی هستند.

البته در برخی موارد می‌توان برای ارسال اشاره‌گر به پروسیجرها از روشهای زیرکانه بهره گرفت. فرض کنید که اولین پارامتر پروسیجر، یک اشاره‌گر به عدد صحیح k باشد. پروسیجر client stub می‌تواند با سازماندهی و تنظیم «مقدار k» در درون یک بسته، آن را برای سرویس‌دهنده بفرستد. پروسیجر server stub پس از ذخیرهٔ مقدار k در حافظه، یک اشاره‌گر برای k ساخته و طبق معمول آن را به پروسیجر سرویس‌دهنده تسلیم می‌کند. وقتی پروسیجر سرویس‌دهنده، کنترل اجرا را به پروسیجر server stub برمی‌گرداند، این پروسیجر مقدار جدید k را برای مشتری باز می‌گرداند تا مقدار قدیم k با مقدار جدیدی که سرویس‌دهنده محاسبه کرده، جایگزین شود. در حقیقت مکانیزم **«فراخوانی به روش ارجاع»**[1] با روش **«فراخوانی از طریق تحویل مقدار و بازیابی نتیجه»** جایگزین شده است. متأسفانه این راهکار همیشه کار نمی‌کند چرا که اگر اشاره‌گر مثلاً به یک گراف یا ساختمان دادهٔ پیچیده اشاره کرده باشد نمی‌توان از این روش بهره گرفت. به همین دلیل باید بر روی پارامترهایی که برای فراخوانی پروسیجرهای راه دور مورد نیاز هستند، محدودیتهایی اعمال کرد.

مسئله دوم آن است که در زبانهایی مثل C که در آن محدودیتهای «نوع داده» (Data Type) چندان قوی نیست نوشتن پروسیجری که مثلاً ضرب داخلی دو بردار (دو آرایه) را حساب می‌کند، بدون مشخص کردن بزرگی اندازهٔ آرایه‌ها، کاملاً صحیح و معتبر است. حال مثلاً قرارداد شده که انتهای این آرایه‌ها با یک مقدار ویژه مشخص گردد و

۱. Call by Reference

پروسیجر صدازننده و صداشونده هر دو آنرا بـه رسـمیت مـی‌شناسند. در چـنین شـرایـطی client stub اسـاساً نمی‌تواند چنین پارامترهایی را استخراج، سازماندهی (مارشالینگ) و ارسال نماید چرا که اندازهٔ پارامترها را نمی‌داند.

مسئله سوم آن است که همیشه این امکان وجود ندارد که بتوان نوع داده‌ای پارامترها را از روی توصیف رسمی (Formal Specification) یا کد برنامه استنتاج کرد. مثلاً printf می‌تواند به هر تعداد پارامتر داشته باشد (حداقل یکی) و پارامترها نیز می‌توانند مخلوطی از متغیرهای صحیح (از نـوع int، short، long)، کـاراکـتر، رشـته‌ای (string)، اعداد اعشاری با طول دلخواه و انواع دیگر باشند. تلاش در فراخوانی پروسیجر printf از راه دور، عملاً غیرممکن است چرا که زبان C در خصوص انواع داده آسان می‌گیرد. از طرفی اگر قانونی وضع شود که استفاده از RPC را منوط به عدم استفاده از C (یا C++) کند، آنرا از عمومیت و رواج می‌اندازد.

مسئله چهارم در ارتباط با متغیرهای سراسری برنامه است. بطور طبیعی پروسیجرهای صدازننده و صداشونده می‌توانند به غیر از پارامترها از طریق متغیرهای سراسری نیز با یکدیگر تبادل داده داشته باشند. اگـر پـروسیجر فراخوانی شده به یک ماشین راه دور منتقل شود، این کد با شکست مواجه خواهد شد چرا که متغیرهای سراسری در آنجا معتبر و قابل استفاده نیستند.

این مسائل بدین معنا نیست که باید از RPC سلب امید کرد. در واقع بطور گسترده‌ای از آن استفاده می‌شود ولیکن باید محدودیتهایی را اعمال کرد تا در عمل بدرستی کار کند.

البته RPC لزوماً از UDP استفاده نمی‌کند ولیکن RPC و UDP زوج متناسبی هستند و عموماً برای RPC از UDP بهره گرفته می‌شود. علیرغم این، وقتی پارامترها یا نتیجهٔ اجرای پروسیجر از اندازهٔ حداکثر یک بستهٔ UDP بیشتر باشد یا وقتی که عملیات مورد تقاضا قابل تکرار نباشند یا تکرار آن، نتایج نامشخصی به همراه داشته باشد ممکن است به جای استفاده از UDP مجبور به تنظیم یک اتصال TCP و ارسال تقاضا از طریق آن باشیم.

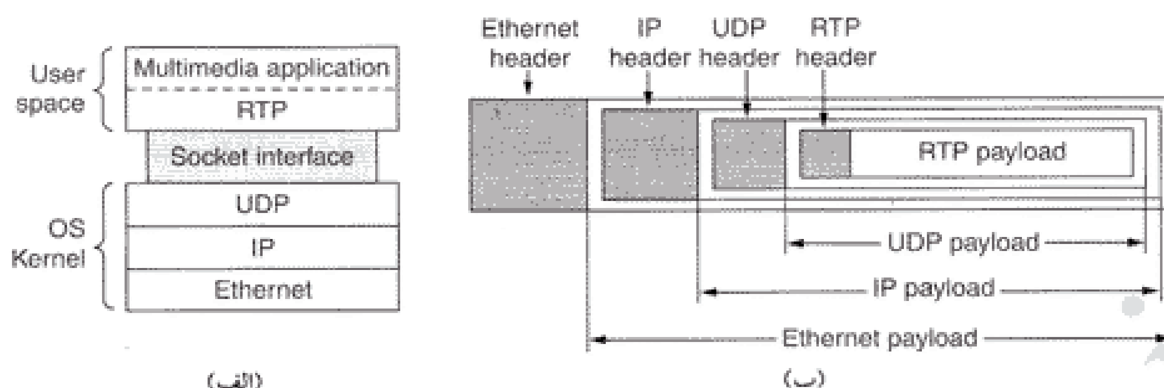
۶-۴-۳ پروتکل انتقال بی‌درنگ

مدل سرویس‌دهنده/مشتری مبتنی RPC، زمینه‌ای است که در آن از UDP استفادهٔ گسترده‌ای می‌شود. زمینهٔ دیگر استفاده از UDP، برنامه‌های کاربردی چند رسانه‌ای بی‌درنگ است. خصوصاً با رواج روزافزون رادیوی اینترنتی، تلفن اینترنتی، موزیک درخواستی از شبکه، ویدیوکنفرانس و دیگر کاربردهای چندرسانه‌ای، عموم افراد دریافتند که همه در حال حرکت به سوی یک پروتکل انتقال بی‌درنگِ کم و بیش مشابه هستند. به تدریج روشن شد که داشتن یک پروتکل عمومی و استاندارد برای انتقال بی‌درنگ داده‌ها ایدهٔ خوبی است. بدین ترتیب پروتکل RTP[1] متولد شد. این پروتکل در سند RFC 1889 تشریح شده و امروزه رواج گسترده‌ای دارد.

جایگاه RTP در پشتهٔ پروتکلی اندکی عجیب به نظر می‌رسد. تصمیم گرفته شد که RTP در فضای پروسهٔ کاربر قرار گرفته و از طریق UDP اجرا شود. عملکرد RTP به نحو زیر است: برنامهٔ چندرسانه‌ای حاوی چندین قطعهٔ صدا، ویدیو، متن و احتمالاً استریمهای دیگر می‌باشد؛ این داده‌ها به «کتابخانهٔ RTP[2]» که در فضای برنامهٔ کاربر قرار دارد و به همراه آن اجرا می‌گردد، خورانده می‌شود. این کتابخانه، استریم داده‌ها را مالتی‌پلکس کرده و آنها را در بسته‌های RPT جاسازی می‌کند. سپس آنها را بر روی یک سوکت قرار می‌دهد. در آنسوی سوکت، (یعنی در هستهٔ سیستم عامل)، بسته‌های UDP تولید شده و در درون بسته‌های IP قرار می‌گیرند. حال اگر کامپیوتر بر روی شبکهٔ اترنت واقع شده باشد، بسته‌های IP جهت انتقال بر روی کانال، در درون یک فریم اترنت جاسازی می‌شود. در چنین شرایطی، پشتهٔ پروتکلی شبیه به شکل ۶-۲۵-الف خواهد بود. تودرتویی بسته‌ها نیز در شکل ۶-۲۵-ب نشان داده شده است.

---

۱. Real-time Transport Protocol
۲. RTP Library



شکل ۶-۲۵. (الف) موقعیت RTP در پشتهٔ پروتکلی. (ب) ترتیب تودرتویی بسته‌ها.

در نتیجهٔ این سبک طراحی ، به سادگی نمی‌توان گفت که RTP در کدام لایه قرار می‌گیرد. از آنجایی که RTP در فضای برنامهٔ کاربر اجرا می‌شود و نهایتاً به برنامهٔ کاربردی لینک می‌شود فلذا به یک پروتکل لایهٔ کاربرد شبیه است. از طرف دیگر، RTP یک پروتکل مستقل از لایهٔ کاربرد است و امکانات لایهٔ انتقال را عرضه می‌کند لذا شبیه به یک پروتکل لایهٔ انتقال به نظر می‌رسد. شاید بهترین تعبیر آن باشد که: «RTP یک پروتکل لایهٔ انتقال است که در لایهٔ کاربرد پیاده‌سازی شده است.»

عملکرد اصلی پروتکل RTP آنست که چندین استریم از داده‌های بی‌درنگ را در یک استریم از بسته‌های UDP مالتی‌پلکس کند. این استریم UDP را می‌توان برای یک ماشین مقصد (یعنی تک‌پخشی) یا چندین ماشین مقصد (یعنی چندپخشی) فرستاد. از آنجایی که RTP صرفاً از بسته‌های معمولی UDP بهره می‌گیرد لذا مسیریابها با این بسته‌ها به صورت ویژه رفتار نمی‌کنند مگر آن که ویژگی کیفیت خدمات (QoS) در IP پیاده و فعّال شود. خصوصاً آنکه هیچ تضمینی در خصوص تحویل این بسته‌ها، میزان تأخیر و لرزش (Jitter) وجود ندارد.

به هر بسته در یک استریم RTP ، شماره‌ای داده می‌شود که یک واحد از شمارهٔ بستهٔ قبلی بیشتر است. شماره‌گذاری بسته‌ها به مقصد اجازه می‌دهد تا گم شدن احتمالی بسته‌ها را تشخیص بدهد. اگر بسته‌ای از دست برود بهترین کاری که مقصد می‌تواند انجام بدهد آنست که مقادیر بستهٔ از دست رفته را به روش درون‌یابی (Interpolation) تخمین بزند.[۱] ارسال مجدد بسته‌های گم‌شده از لحاظ عملی ممکن نیست زیرا بسته‌هایی که مجدداً ارسال می‌شوند، احتمالاً آنقدر دیر به مقصد می‌رسند که دریافت آنها هیچ فایده‌ای نخواهد داشت. در نتیجه پروتکل RTP هیچ مکانیزمی برای کنترل جریان، کنترل خطا، اعلام وصول بسته (Ack) یا تقاضای ارسال مجدد ندارد.

محتوای بسته‌های RTP می‌تواند انواع مختلف داده را در بربگیرد و طبعاً به دلخواه برنامهٔ کاربردی کُد می‌شود. برای جلوگیری از تنوع زیاد و ناسازگاری، RTP چندین پروفایل (مثل استریم صدا) تعریف و برای هر پروفایل چندین روش کدینگ پیشنهاد کرده است. به عنوان مثال یک استریم صدا می‌تواند به روش PCM هشت بیتی با نرخ نمونه‌برداری 8KHz یا روش «مدولاسیون دلتا[۲]» ، «کدینگ پیشگویانه[۳]» ، «کدینگ GSM»، روش MP3 یا نظائر آن کُد و ارسال شود. پروتکل RTP فیلدی را در سرآیند هر بسته پیش‌بینی کرده تا به کمک آن مبداء بتواند

۱. درون‌یابی بسته‌های گمشده که برای صدا و تصاویر ویدیویی کاربرد دارد بدین نحو است که گیرنده از روی فریمهای قبلی و بعدی بخشی از داده‌های از دست رفته را بصورت تقریبی محاسبه و آنرا جایگزین می‌کند تا کیفیت تصویر غیرقابل تحمل نشود.

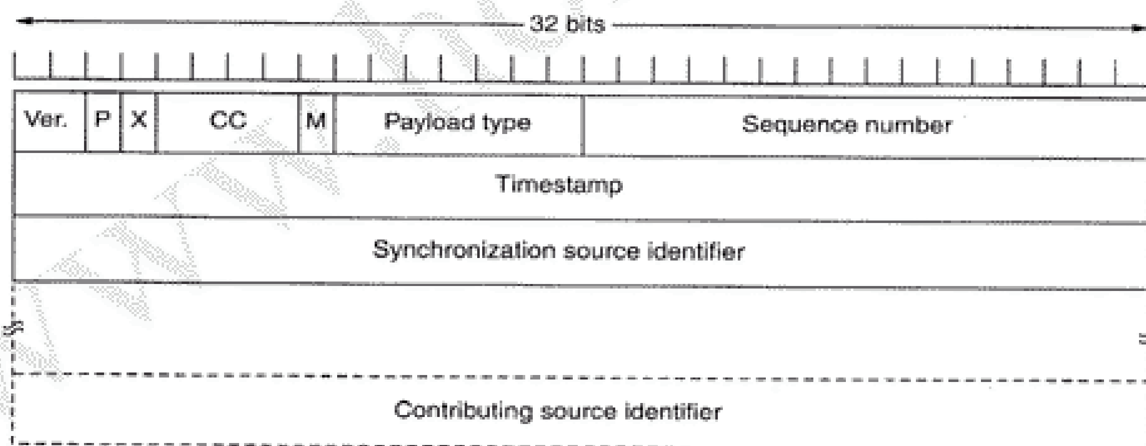
۲. Delta Modulation
۳. Predictive Encoding

نوع کدینگ خود را مشخص کند.

امکان دیگری که بسیاری از کاربردهای بی‌درنگ بدان نیاز دارند، «درج مُهر زمان» (Timestamping) برروی بسته‌هاست. ایدهٔ اصلی آنست که به مبداء اجازه بدهیم تا به اولین نمونه[1] از هر بسته یک مهر زمان نسبت بدهد. مُهر زمان نسبت به زمان شروع استریم محاسبه می‌شود فلذا فقط اختلاف مقادیر مُهر زمان بسته‌ها اهمیت دارد و مقادیر مطلق آنها بی‌ارزش است. در این مکانیزم، مقصد نیاز کمی به بافرسازی بسته‌ها خواهد داشت و فارغ از آن که بسته‌ها چه زمانی دریافت شده‌اند، براساس مهر زمان درج شده بر روی آنها، سر موعد مقرر اجرا (Play) می‌شوند.[2] درج مهر زمان نه تنها تاثیر منفی «لرزش» (Jitter) را کاهش می‌دهد بلکه امکان آنرا فراهم می‌آورد تا استریمها از لحاظ زمانی سنکرون بمانند. به عنوان مثال یک برنامهٔ تلویزیون دیجیتالی می‌تواند یک استریم ویدیو و دو استریم صدا داشته باشد. این دو استریم صدا می‌توانند یکی برای پخش فیلم با صدای اصلی و دیگری با صدای دوبله شده به زبان محلی باشد تا بیننده به میل خود یکی از آنها را انتخاب کند. هر یک از این استریمها از ابزارهای فیزیکی متفاوتی منشاء می‌گیرند ولیکن اگر با یک شمارندهٔ واحد شماره‌گذاری شوند، می‌توان آنها را صورت هماهنگ و سنکرون اجرا کرد، حتی اگر به صورت نامنظم و پس و پیش ارسال شده باشند.

سرآیند بستهٔ RTP در شکل ۶-۲۶ نشان داده شده است. این سرآیند از سه کلمهٔ ۳۲ بیتی و چند بخش اختیاری توسعه (Extension) تشکیل شده است. اولین کلمه، در برگیرندهٔ این فیلدهاست: فیلد Version که فعلاً ۲ است. امیدواریم که این نسخه نزدیک به نسخهٔ متکامل و نهایی آن باشد چراکه فقط یک شمارهٔ بیشتر (یعنی نسخهٔ ۳) باقی نمانده است. (البته می‌توان عدد ۳ را بدین نحو تعریف و تعبیر کرد که شمارهٔ واقعی نسخهٔ پروتکل درون سرآیند اختیاری قرار گرفته است!)

بیت **P**، بیانگر آنست که به بسته، داده‌های زائدی اضافه شده تا طول آن ضریبی از ۴ باشد. در حقیقت مقدار آخرین بایت زائد مشخص می‌کند که چند بایت به داده‌ها اضافه شده است.[3] بیت **X** مشخص می‌کند که در این بسته



شکل ۶-۲۶. سرآیند RTP (RTP Header).

---

۱. sample

۲. در ذهن خود، «مُهر زمان» را برای صدا (یا ویدیو) بدینگونه تصور کنید: اولین نمونهٔ صدا در لحظهٔ $t=t_0$ تولید شده است. بنابراین فرضاً در مهر زمان اولین بستهٔ ارسالی ۰ درج می‌شود. زمان تولید بسته‌های بعدی نسبت به اولین بسته، محاسبه و درون فیلد مهر زمان درج می‌شود. بنابراین گیرنده حتی اگر بسته‌ها را بصورت نامنظم دریافت کند براساس این مقدار می‌تواند زمان دقیق اجرای هر بسته را مشخص نماید. -م

۳. بعبارتی بیت P مشخص می‌کند که حداقل یک بایت زائد به داده‌ها اضافه شده است. آخرین بایت، تعداد بایتهای زائد را مشخص می‌کند. -م

«سرآیند توسعه» (Extension Header) وجود دارد. قالب و معنای سرآیند توسعه در پروتکل تعریف نشده است جز آنکه اولین کلمهٔ سرآیند توسعه، طول آنرا مشخص می‌کند. این سرآیند امکان آنرا فراهم آورده تا بتوان نیازهای پیش‌بینی نشده را مرتفع کرد.

فیلد CC بیانگر آنست که چند مبداء در تولید بسته دخالت داشته‌اند (از صفر تا ۱۵). در ادامه به این موضوع خواهیم پرداخت. بیت M یک علامت ویژهٔ برنامه‌های کاربردی است. مثلاً می‌توان از این علامت به معنای شروع اولین فریم ویدیو، شروع اولین کلمه در کانال صوتی (یا هر چه که برنامهٔ کاربردی بخواهد) استفاده کرد.

فیلد Payload Type روش کدینگ مورد استفاده را تعیین می‌کند (به عنوان مثال صدای هشت بیتی فشرده‌نشده، MP3 یا امثال آن). از آنجایی که هر بسته، این فیلد را با خود حمل می‌کند فلذا می‌توان در حین ارسال روش کُدینگ را تغییر داد.

فیلد Sequence Number (شمارهٔ ترتیب) یک شمارنده است که به ازای ارسال هر بستهٔ RTP یک واحد افزایش می‌یابد. از این شماره برای کشف بسته‌های گمشده استفاده می‌شود. در فیلد Timestamp ، «مُهر زمان» درج می‌شود تا مشخص گردد بستهٔ جاری (نسبت به زمان تولید اولین بسته) دقیقاً در چه زمانی تولید شده است.[1] با درج مهر زمان، «لرزش» (Jitter) کاهش می‌یابد زیرا لحظهٔ اجرای[2] هر بسته از زمان دریافت بسته جدا می‌شود. فیلد Synchronization Source Identifier مشخص می‌کند که بستهٔ جاری به کدام استریم تعلق دارد.[3] به کمک این فیلد استریمهای متعدد در یک دنباله از بسته‌های UDP معمولی، مالتی‌پلکس یا دی‌مالتی‌پلکس می‌شوند.

در آخر، فیلد Contributing Source Identifier قرار می‌گیرد که در صورت استفاده، بیانگر آنست که در استودیو، میکسر وجود دارد.[4] در چنین حالتی، میکسر موظف به سنکرونیزاسیون استریمهاست؛ در ضمن استریمهایی که باید میکس گردند در این فیلد فهرست می‌شوند.

پروتکل RTP یک خواهر کوچک به نام RTCP[5] دارد. (شاید هم دوقلو و همزاد باشند!) این پروتکل عملیات فیدبک، سنکرون‌سازی و واسط کاربری را مدیریت می‌کند ولیکن هیچ داده‌ای را انتقال نمی‌دهد. اولین کاربرد آن گرفتن فیدبک از میزان تأخیر، لرزش (Jitter)، پهنای باند، ازدحام (congestion) و کسب آگاهی از دیگر ویژگیهای شبکه برای ماشین مبداء است. از این اطلاعات می‌توان در فرآیند کدینگ داده‌ها بهره گرفت تا مثلاً وقتی که شبکه خوب عمل می‌کند، نرخ ارسال را افزایش بدهد (یا کیفیت را بهبود ببخشد) و هنگامی که مشکلی در شبکه حادث می‌شود (مثل ازدحام) نرخ ارسال را پایین بیاورد. با در اختیار داشتن یک فیدبک دائم از شرایط فعلی شبکه، الگوریتم کدینگ می‌تواند همیشه بهترین روش را در تناسب با شرایط جاری، انتخاب نماید. به عنوان مثال وقتی پهنای باند در حین ارسال افزایش یا کاهش می‌یابد، الگوریتم کدینگ می‌تواند از روش MP3 به PCM (یا بالعکس) تغییر روش بدهد. وجود فیلد Payload Type این امکان را فراهم آورده که در هر لحظه از زمان بتوان الگوریتم کدینگ را به روش مناسب تغییر داد.

پروتکل RTCP ، سنکرون سازی بین استریمها را نیز بر عهده دارد. مشکل از آنجا ناشی می‌شود که

---

۱. دقت کنید که این زمان مربوط به تولید اولین کلمهٔ بسته یا به عبارت دقیقتر اولین نمونه (Sample) در بسته است زیرا بسته مملو از کلمات یا نمونه‌های مربوط به صدا، تصویر یا امثالهم است که از لحاظ زمانی بعد از اولین نمونه تولید شده‌اند. ـم

۲. Playback time

۳. استریمهای صدا، تصویر و امثالهم می‌توانند بطور همزمان تولید یا دریافت شوند لذا باید راهی برای تفکیک و سنکرونیزاسیون بسته‌های متعلق به هر استریم وجود داشته باشد. ـم

۴. بدین معنا که همانند یک استودیو، منابع تولید استریم بسیار متعددند و نهایتاً باید میکس شوند. ـم

۵. Real-time Transport Control Protocol

استریمهای متفاوت از سیگنالهای ساعت با مشخصات متفاوتی مثل فرکانس، Drift بهره می‌گیرند. پروتکل RTCP می‌تواند برای سنکرون‌سازی این استریمها بکار گرفته شود.

در آخر، RTCP روشی برای نامگذاری مبداء استریمهای مختلف (مثلاً در قالب اسامی ASCII) ارائه کرده است. این اطلاعات می‌تواند بر روی صفحهٔ نمایش گیرنده، نشان داده شود تا مشخص شود چه کسی در حال صحبت یا ارسال استریم است.

اطلاعات بیشتر در خصوص RTP را می‌توانید در مرجع (Perkins, 2002) بیابید.

۶ـ۵ پروتکلهای لایهٔ انتقال در اینترنت: TCP[1]

UDP پروتکل بسیار ساده‌ای است و کاربردهای خاص خود را (مثل تعامل سرویس‌دهنده/مشتری یا کاربردهای چند رسانه‌ای) دارد ولیکن اغلب کاربردهای اینترنت به تحویل مطمئن و مرتب شدهٔ داده‌ها نیازمندند. UDP چنین امکانی را فراهم نمی‌کند و طبعاً به پروتکل دیگری احتیاج است. این پروتکل TCP نام دارد و بیشتر بار اینترنت را به دوش می‌کشد. اجازه بدهید جزئیات این پروتکل را بررسی نماییم.

۱ـ۵ـ۶ مقدمه‌ای بر TCP

TCP بدین منظور طراحی شد تا یک دنباله (استریم) از بایتها را به صورت مطمئن و عاری از خطا بین دو نقطهٔ پایانی (یعنی دو پروسه) از شبکه‌ای که نامطمئن و مستعد خطاست، منتقل نماید. «شبکه‌ای از شبکه‌ها» (Internetwork) از لحاظ ساختاری با یک شبکهٔ واحد تفاوت دارد زیرا هر بخش از آن دارای توپولوژی مختلف، پهنای باند، تأخیر، طول بسته و پارامترهای متفاوت است. TCP طراحی شد تا بطور پویا خود را با ویژگیهای چنین شبکهٔ ناهمگونی تطبیق بدهد و در مقابل ناکارآمدیها و شکستهای گوناگون، تحمل‌پذیر (Robust) باشد.

پروتکل TCP رسماً در RFC 793 تعریف شده است. به مرور زمان، انواع خطاها و تناقضات آن آشکار شد و در برخی از زمینه‌ها، نیازها نیز تغییر کرد. تبیین این مفاد و برخی از اصلاحیه‌ها در RFC 1122 آمده است. بسط و بهبود آن نیز در RFC 1323 تشریح شده است.

هر ماشین که از TCP پشتیبانی می‌کند باید دارای **«واحد انتقال TCP»**[2] باشد، خواه در قالب پروسیجرهای کتابخانه‌ای یا یک پروسهٔ کاربری و خواه در قالب بخشی از هستهٔ سیستم عامل. در تمامی این حالات، واحد انتقال، استریمهای TCP را مدیریت کرده و به صورت واسطی بین لایهٔ IP و پروسه‌های کاربردی انجام وظیفه می‌کند. «واحد انتقال» دنباله‌ای از داده‌های کاربر را از پروسه‌های محلی گرفته و آن را به قطعاتی با طول کمتر از 64KB می‌شکند (البته در عمل این قطعات اغلب ۱۴۶۰ بایتی هستند تا پس از الحاق سرآیند IP و TCP به آنها، جمعاً ۱۵۰۰ بایت شده و متناسب با ظرفیت یک فریم اترنت باشند.) سپس هر یک از این قطعات را [پس از افزودن سرآیند لازم] در قالب یک دیتاگرام مستقل IP ارسال می‌نماید. وقتی دیتاگرامهای حاوی داده‌های TCP به ماشین مقصد می‌رسند، به واحد انتقال تحویل داده شده و براساس آنها دنبالهٔ بایتهای اصلی ساخته خواهد شد. برای سادگی اغلب بجای «واحد انتقال TCP» (یعنی یک قطعه نرم‌افزار) و هم بجای پروتکل TCP (یعنی مجموعه‌ای از قوانین) به تنهایی از واژهٔ TCP استفاده می‌نمائیم. مضمون کلام مشخص می‌کند که منظورمان از TCP چیست. مثلاً وقتی می‌گوئیم «کاربر داده‌ها را به TCP تحویل می‌دهد» روشن است که منظور از TCP همان «واحد انتقال TCP» است.

لایهٔ IP هیچ تضمینی در تحویل صحیح و مرتب بسته‌ها نمی‌دهد لذا این وظیفه بر عهدهٔ TCP است که پس از

---

۱. Transmission Control Protocol

۲. TCP Transport Entity

انقضای مهلت مقرر آنها را از نو ارسال نماید. از طرفی ممکن است دیتاگرامها به صورت نامرتب و پس‌وپیش به مقصد برسند لذا وظیفهٔ دیگر TCP آن است که آنها را مرتب کرده و پیام اصلی را بازسازی نماید. کوتاه سخن آنکه TCP باید آن قابلیت اطمینانی را که کاربران می‌خواهند ولی IP عرضه نمی‌کند، در اختیار آنان قرار بدهد.

۶-۵-۲ مدل خدمات TCP

برای استفاده از خدمات TCP، پروسه‌های گیرنده و فرستنده یک نقطهٔ پایانی به نام «**سوکت**» ایجاد می‌کنند. (در مورد سوکت در بخش ۶-۱-۳ بحث شد.) هر سوکت دارای یک «شمارهٔ سوکت» (یا به عبارتی آدرس سوکت) است که این شماره متشکل از آدرس IP ماشین میزبان و یک عدد ۱۶ بیتی یکتا و محلی است که اصطلاحاً «**پورت**» (Port) نامیده می‌شود. «پورت» معادل نام TSAP در ادبیات پروتکل TCP است. برای بهره‌گیری از خدمات TCP باید یک اتصال بین سوکت ماشین مبداء و سوکت ماشین مقصد ایجاد شود. توابع اولیهٔ مرتبط با سوکت در شکل ۶-۵ فهرست شده‌اند.

می‌توان از یک سوکت برای برقراری چندین اتصال همزمان بهره گرفت. به عبارت دیگر ممکن است دو یا چند اتصال به یک سوکت واحد ختم شوند. هویت هر «اتصال» توسط شناسه‌های سوکت طرفین در قالب (*socket1* , *socket2*) مشخص می‌گردد. یعنی از شماره‌های مدار مجازی یا هر شناسهٔ دیگر استفاده نمی‌شود.

پورتهائی با شمارهٔ زیر ۱۰۲۴ اصطلاحاً به نام «**پورتهای شناخته‌شده**»[1] مشهورند و برای سرویسهای استاندارد رزرو شده‌اند. مثلاً هر پروسه‌ای که می‌خواهد به منظور انتقال فایل، با ماشینی ایجاد اتصال کند می‌تواند با پورت ۲۱ از ماشین مقصد که «دیمون FTP» (FTP Daemon) بدان گوش می‌کند تماس بگیرد. فهرست پورتهای شناخته‌شده در آدرس www.iana.com در دسترس می‌باشد. تاکنون بیش از ۳۰۰ شماره از آنها منتسب شده‌اند که چند تا از مشهورترین شمارهٔ پورتها را در شکل ۶-۲۷ ملاحظه می‌کنید.

در هنگام راه‌اندازی سیستم، دیمون FTP می‌تواند خود را به پورت ۲۱ متصل کند؛ دیمون TelNet به پورت ۲۳ و دیگر پروسه‌ها نیز به همین ترتیب می‌توانند خود را به یک پورت متصل نمایند. از آنجایی که اکثر این دیمونها در بیشتر زمانها بیکار هستند لذا اجرای آنها در هنگام راه‌اندازی سیستم، فضای حافظه را بیهوده تلف خواهد کرد. به جای همهٔ آنها، عموماً یک دیمون واحد که در یونیکس به «دیمون inetd»[2] مشهور است، اجرا شده و بطور همزمان خود را به چندین پورت متصل می‌نماید و منتظر تماسهای ورودی می‌شود. وقتی تقاضای برقراری یک اتصال دریافت شد، پروسهٔ inetd یک پروسهٔ جدید ایجاد کرده و دیمون مناسب را برای سرویس‌دهی به این تقاضا، راه اندازی و اجرا می‌نماید. بدین نحو، تمام دیمونهای دیگر به غیر از دیمون inetd فقط و فقط زمانی فعال می‌شوند که کاری برای انجام داشته باشند. دیمون inetd شمارهٔ پورتهایی که باید به آنها گوش بدهد را از فایل پیکربندی خاصّ خود استخراج می‌نماید. مسئول سیستم (admin) می‌تواند سیستم را به گونه‌ای پیکربندی کند که برخی از دایمونها به طور دائم به یک پورت گوش بدهند (مثل پورت ۸۰) و به مابقی پورتها فقط دایمون inetd گوش بدهد.

تمام اتصالات TCP، «دوطرفهٔ کامل»[3] و «نقطه‌به‌نقطه» هستند. «دو طرفهٔ کامل» یعنی ترافیک می‌تواند بطور همزمان در دو جهت جریان داشته باشد؛ نقطه‌به‌نقطه نیز یعنی یک اتصال صرفاً دو نقطهٔ پایانی (End Point) دارد.[4] TCP از ارسال چند پخشی (Multicasting) یا پخش فراگیر (Broadcast) پشتیبانی نمی‌کند.

یک اتصال TCP استریمی از بایتها را منتقل می‌نماید نه دنباله‌ای از پیامها را؛ یعنی مرز پیامهایی که بین دو نقطه

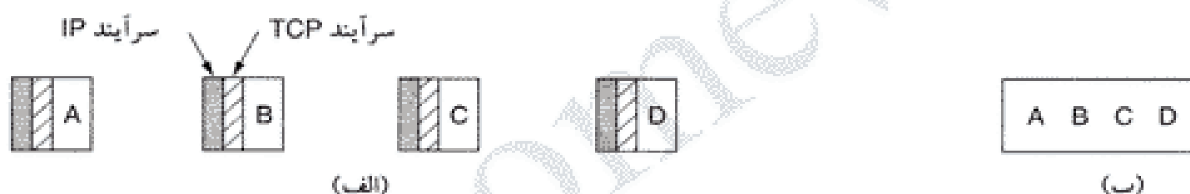
---

۱. Well-known Ports
۲. Internet Daemon
۳. Full Duplex و Point-to-Point
۴. یا بعبارتی صرفاً دو پروسهٔ مبداء و مقصد با یکدیگر محاوره می‌کنند. -م

پورت	پروتکل	کاربرد
21	FTP	انتقال فایل
23	Telnet	ورود به سیستم از راه دور
25	SMTP	پست الکترونیکی (ایمیل)
69	TFTP	پروتکل ساده انتقال فایل
79	Finger	جستجوی اطلاعات در خصوص یک کاربر
80	HTTP	تور جهان‌گستر یا همان سیستم جهانی وب
110	POP-3	دسترسی به نامه‌های الکترونیکی از راه دور
119	NNTP	اخبار یوزنت (سیستم خبررسان)

شکل ۶-۲۷. برخی از شماره‌های پورت انتساب داده شدهٔ مشهور.

مبادله می‌شوند حفظ نخواهد شد. به عنوان مثال اگر پروسهٔ فرستنده، چهار بستهٔ ۵۱۲ بایتی را بر روی یک استریم بفرستد ممکن است در قالب چهار قطعه دادهٔ ۵۱۲ بایتی یا دو قطعهٔ ۱۰۲۴ بایتی یا یک قطعه ۲۰۴۸ یا بترتیبی مشابه، به پروسهٔ گیرنده تحویل شود. (شکل ۶-۲۸ را ببینید.) هیچ راهی برای آن که گیرنده بتواند اندازهٔ قطعات داده را تشخیص بدهد، وجود ندارد.



شکل ۶-۲۸. (الف) چهار قطعهٔ ۵۱۲ بایتی در قالب دیتاگرامهای مستقل IP ارسال شده‌اند. (ب) تودهٔ ۲۰۴۸ بایتی داده‌ها، بطور یکجا و با یکبار فراخوانی READ، تحویل برنامهٔ کاربردی شده است.

در یونیکس، فایلها نیز همین ویژگی را دارند یعنی پروسه‌ای که از یک فایل می‌خواند نمی‌تواند بفهمد که آیا فایل به صورت بلوکی نوشته شده، بایت به بایت نوشته شده یا به صورت یکجا و در آن واحد نوشته شده است. همانند یک فایل در یونیکس، نرم‌افزار TCP نیز هیچ تصوری از معنای بایتها ندارد. یک بایت فقط یک بایت است!

وقتی یک برنامهٔ کاربردی داده‌های خود را به TCP تحویل می‌دهد، ممکن است TCP آنها را فوراً ارسال نماید یا آنها را موقتاً بافر کند. با این حال برخی از برنامه‌های کاربردی می‌خواهند که داده‌هایشان فوراً ارسال شود. مثلاً فرض کنید کاربری از راه دور به ماشینی وارد شده باشد (عمل login). پس از آنکه خط فرمان تکمیل و کلید Enter زده شد، انتظار می‌رود که این خط فوراً به ماشین راه دور تحویل شود و بهیچوجه تا رسیدن خط فرمان بعدی بافر نگردد. برای آن که TCP مجبور به ارسال سریع و آنی داده‌ها شود، برنامهٔ کاربردی می‌تواند از بیت پرچم PUSH (PUSH Flag) بهره بگیرد تا به TCP تفهیم شود که انتقال داده‌ها نباید به تأخیر بیفتد. (در خصوص این بیت بعداً صحبت خواهیم کرد.)

برخی از برنامه‌های کاربردی ابتدایی، از بیت پرچم PUSH به عنوان نوعی نشانه برای تعیین مرز هر پیام بهره می‌گرفتند.[1] اگرچه این حقّه گاهی اوقات کار می‌کند ولی گاهی نیز با شکست مواجه می‌شود زیرا در تمام

---

۱. به عبارت دیگر وقتی یک بستهٔ TCP که در آن بیت پرچم PUSH فعال شده، می‌رسد اگرچه بدین معناست که نباید بافر شود و سریعاً باید به برنامهٔ کاربردی تحویل گردد ولی برخی برنامه‌های کاربردی ابتدایی از این بیت به عنوان حقه‌ای جهت تفهیم پایان پیام جاری به طرف مقابل، بهره می‌گرفتند. -م

پیاده‌سازیهای TCP، در طرف گیرنده الزاماً بیت پرچم PUSH به برنامهٔ کاربردی تحویل داده نمی‌شود. مضاف براین، اگر قبل از آنکه اولین بسته با پرچم PUSH مَجال ارسال پیدا کند (مثلاً به دلیل شلوغی خط خروجی)، چند بستهٔ دیگر از همین نوع تولید و تحویل TCP شود، TCP مُجاز خواهد بود که تمام بسته‌های با علامت PUSH را پشت سرهم ادغام کرده و آن را در قالب یک دیتاگرام IP به صورت یکجا ارسال نماید که در این صورت مرز قطعات مختلف قابل تشخیص نخواهد بود.

یکی دیگر از خدمات TCP که اشاره به آن خالی از لطف نخواهد بود، موضوع **«داده‌های اضطراری»** (Urgent Data) است. وقتی کاربری که از راه دور با یک برنامه‌ای کاربردی در تعامل است کلید DEL یا CTRL-C را فشار می‌دهد (تا روند اجرای برنامه‌ای که از قبل شروع شده را قطع کند)، برنامهٔ کاربردی فرستنده، پاره‌ای اطلاعات کنترلی در «استریم داده» قرار داده و بیت پرچم URGENT را در آن فعّال و آنرا جهت ارسال به TCP تسلیم می‌کند. این رخداد باعث می‌شود که TCP به جمع‌آوری داده خاتمه بدهد و هر آنچه راکه دارد سریعاً برای طرف مقابل بفرستد.

وقتی «داده‌های اضطراری» به مقصد می‌رسند به برنامهٔ کاربردی گیرنده، «وقفه» داده می‌شود (در اصطلاح یونیکس به آن برنامه سیگنال داده می‌شود)؛ آن برنامه طبعاً کارش را متوقف کرده و استریم داده را می‌خواند تا داده‌های اضطراری را یافته و سریعاً پردازش نماید. پایان داده‌های اضطراری علامت‌گذاری شده است لذا برنامهٔ کاربردی به راحتی می‌تواند انتهای این داده‌ها را تشخیص بدهد. البته ابتدا داده‌های اضطراری علامتگذاری نمی‌شود و تشخیص آن بر عهدهٔ برنامهٔ کاربردی است. این روش فقط مکانیزم سیگنال‌دهی به پروسه‌ها را ارائه می‌کند و مدیریت بقیهٔ امور بر عهدهٔ برنامهٔ کاربردی گذاشته شده است.

۶-۵-۳ پروتکل TCP

در این بخش یک دید کلی از پروتکل TCP ارائه خواهیم کرد و در بخش بعدی سرآیند پروتکل را فیلد به فیلد بررسی می‌نماییم.

ویژگی کلیدی در پروتکل TCP (که طراحی کل پروتکل از آن تاثیر پذیرفته) آنست که هر بایت ارسالی بر روی یک اتصال TCP دارای یک شمارهٔ ترتیب ۳۲بیتی است. زمانی که اینترنت شروع به کار کرد، خطوط ارتباطی بین مسیریابها، اغلب خطوط اجاره‌ای 56kbps بودند و حتی اگر یک ماشین با تمام سرعت، اقدام به ارسال داده می‌کرد بیش از یک هفته طول می‌کشید تا این شمارهٔ ۳۲بیتی به آخر رسیده و به صفر برگردد. در سرعتهای جدید شبکه، شماره‌های ترتیب می‌توانند در مدت کوتاه و خطرناکی به آخر رسیده و تکرار شوند. (بعداً به این مسئله بیشتر می‌پردازیم.) برای اعلام وصول داده‌ها (یعنی Acknowledgement) و «مکانیزم پنجره» نیز از فیلدها و شماره‌های ترتیب متفاوتی استفاده شده است.

واحدهای انتقال TCP در سمت فرستنده و گیرنده، داده‌ها را در قالب یکسری **«قطعه»** (Segment) ردّ و بدل می‌کنند. هر قطعهٔ TCP متشکل از ۲۰ بایت سرآیند ثابت و اجباری (و در صورت نیاز یک بخش اختیاری در سرآیند) است و به دنبال آن به تعداد صفر یا چند بایت داده قرار می‌گیرد. نرم‌افزار TCP راساً در مورد اندازهٔ هر قطعه تصمیم می‌گیرد.[۱] TCP ممکن است داده‌هایی را که در چند مرحله جهت ارسال در استریم نوشته شده‌اند، جمع‌آوری کرده و آنها را در قالب یک قطعهٔ TCP بفرستد و یا بالعکس داده‌هایی را که در یک مرحله تحویل می‌شوند ، در چند قطعه ارسال نماید. دو عامل می‌تواند محدودکنندهٔ اندازهٔ قطعات باشد. اول آن که هر قطعه

۱. یعنی برنامهٔ کاربردی نمی‌تواند در خصوص آنکه بسته‌ها (قطعه‌ها) در چه اندازه‌ای ارسال شود، اعمال نظر کند. -م

(شامل سرآیند آن) باید بتواند در فضای ۶۵۵۱۵ بایتی فیلد داده از بستهٔ IP جا بگیرد؛ دوم آن که در هر شبکه پارامتری به نام MTU (Maximum Transfer Unit) (یعنی حداکثر طول بستهٔ قابل انتقال) وجود دارد و اندازهٔ هر بسته باید متناسب با این مقدار باشد. در عمل، MTU عموماً ۱۵۰۰ بایت است (اندازهٔ فیلد حمل دادهٔ اترنت) و طبعاً حد بالایی طول هر قطعه باید متناسب با این مقدار باشد.

TCP از پروتکل «پنجرهٔ لغزان» بهره گرفته است: به محض آن که فرستنده، قطعه‌ای را ارسال می‌کند یک تایمر برای آن روشن می‌نماید. وقتی این قطعه به مقصد رسید، واحد TCP (TCP Entity)، قطعه‌ای را بر می‌گرداند که در آن «شمارهٔ تصدیق بستهٔ دریافتی» (Ack.No.) درج شده است. این قطعه می‌تواند حاوی داده‌های متقابل گیرنده باشد و در صورت عدم وجود هر گونه داده، بدون داده ارسال شود. شمارهٔ تصدیق بستهٔ دریافتی در حقیقت حاوی شمارهٔ ترتیب بایتی است که گیرنده از آن شماره به بعد منتظر دریافت آنهاست. اگر مهلت تایمر به پایان برسد و در این زمان دریافت قطعهٔ ارسالی، تصدیق نشود، فرستنده، قطعهٔ قبلی را از نو ارسال می‌کند.

اگرچه این پروتکل ساده به نظر می‌رسد ولی پیچیدگیها و ظرافتهایی چند، در آن نهفته است که در ادامه به آن خواهیم پرداخت. قطعات ارسالی می‌توانند به صورت نامرتب به گیرنده برسند؛ مثلاً اگر بایتهای ۳۰۷۲ تا ۴۰۹۵ در قالب یک قطعه برسد نمی‌توان آن را اعلام وصول کرد چراکه فرضاً بایتهای ۲۰۴۸ تا ۳۰۷۱ نرسیده‌اند. همچنین ممکن است قطعات ارسالی آنقدر در شبکه معطل شوند که مهلت فرستنده منقضی شده و آنها را از نو ارسال نماید و منجر به تولید قطعات تکراری شود. همچنین ارسال مجدد داده‌ها ممکن است در قالب قطعات جدیدی صورت بگیرد.[1] فلذا این موضوع نیز باید به دقت مدیریت و نظارت شود تا گیرنده بتواند بر دنبالهٔ بایتهایی که دریافت کرده و آنهایی که دریافت نکرده به دقت کنترل داشته باشد. خوشبختانه چون هر بایت در یک دنباله (استریم) دارای یک آفست یکتاست فلذا انجام چنین کار مهمی امکان‌پذیر است.

TCP باید بتواند تمام این مسائل را به نحو کارآمد و مؤثری حل و فصل نماید. در ضمن تلاشهای وافری برای بهینه‌سازی کارآیی استریمهای TCP صورت گرفته است تا در مواجهه با مشکلات شبکه خود را تطبیق بدهد. تعدادی از این الگوریتمها که در اغلب پیاده‌سازیهای عملی TCP بکار رفته را در ادامه معرفی کرده‌ایم.

۶-۵-۴ سرآیند قطعهٔ TCP

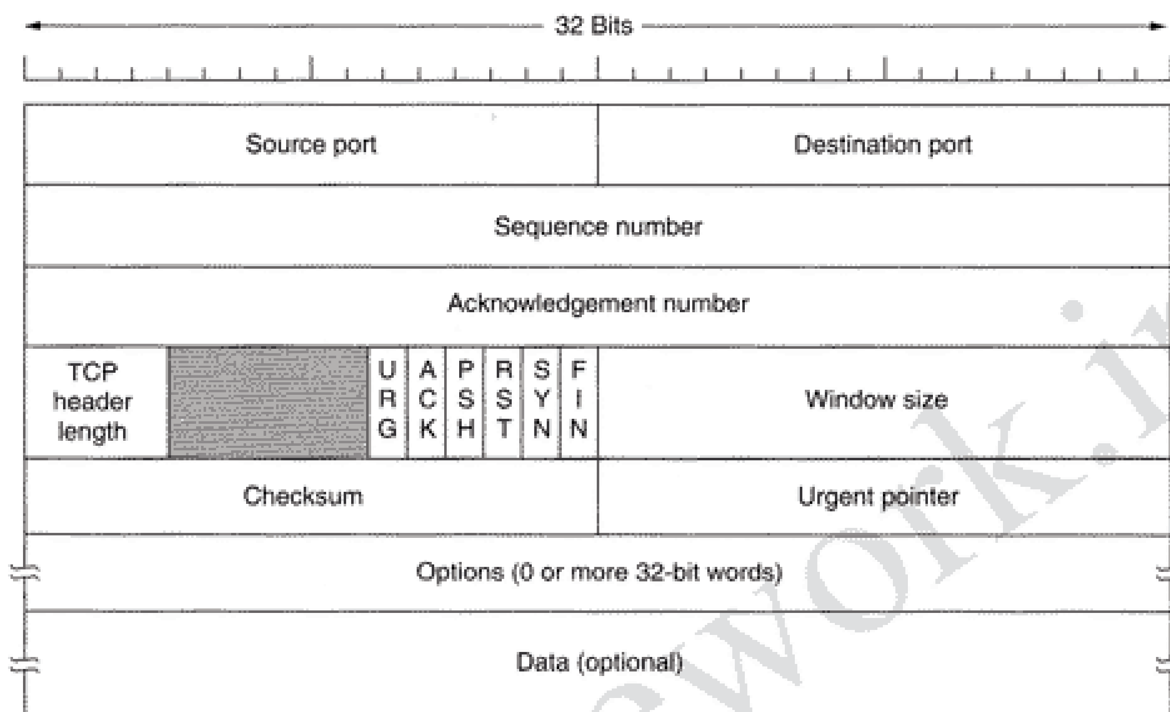
شکل ۶-۲۹، ساختار یک قطعهٔ TCP (TCP Segment) را نشان می‌دهد. هر قطعه با یک سرآیند ۲۰ بایتی و با قالب ثابت شروع می‌شود. پس از این سرآیند ثابت، ممکن است یک سرآیند اختیاری قرار بگیرد. در صورت عدم وجود سرآیند اختیاری، در ادامه می‌تواند حداکثر ۶۵۴۹۵=(۲۰-۲۰-۶۵۵۳۵) بایت داده قرار بگیرد که کسر بیست بایت اول مربوط به سرآیند IP و بیست بایت دوم مربوط به سرآیند TCP است. قطعات بدون داده [یعنی فقط سرآیند]، معتبر و قانونی هستند و عموماً برای اعلام وصول داده‌ها (Ack) و پیامهای کنترلی کاربرد دارند. حال اجازه بدهید که سرآیند TCP را فیلد به فیلد کالبدشکافی کنیم.

فیلدهای «پورت مبداء» (Source Port) و «پورت مقصد» (Destination Port) نقاط انتهایی دو طرف یک اتصال را مشخص می‌نمایند.[2] شمارهٔ پورتهای شناخته‌شده و مشهور در آدرس www.iana.org فهرست شده‌اند ولی هر ماشین میزبان می‌تواند به دلخواه خود، آنها را به پروسه‌ها اختصاص بدهد. یک شمارهٔ پورت ۱۶ بیتی به همراه آدرس IP ماشین میزبان، آدرس ۴۸ بیتی یک نقطهٔ پایانی را تشکیل می‌دهد. آدرس نقاط پایانی مبداء و

---

۱. یعنی مثلاً یک قطعه حاوی ۱۵۰۰ بایت ارسال شده ولی چون وصول آن تصدیق نشده ممکن است ارسال مجدد آن در قالب دو قطعهٔ ۱۰۰۰ و ۵۰۰ بایتی انجام شود. -م

۲. یعنی این دو فیلد هویت پروسه‌های گیرنده و فرستنده را تعیین می‌نمایند. -م



شکل ۶-۲۹. سرآیند TCP.

مقصد، هویت یک اتصال را تبیین می‌کنند.

فیلدهای «شمارهٔ ترتیب» (Sequence Number) و «شمارهٔ تصدیق» (Acknowledgement Number) عملکرد طبیعی خود را دارند یعنی اولی شمارهٔ ترتیب قطعهٔ داده و دومی اعلام وصول داده‌ها است. دقت کنید که فیلد «شمارهٔ تصدیق»، شمارهٔ بایتی را مشخص می‌کند که از آن بایت به بعد منتظر دریافت داده‌هاست نه آخرین بایت دریافتی.[1] هر دوی این فیلدها ۳۲ بیتی هستند زیرا در استریم TCP هر بایت دارای شمارهٔ ترتیب است.

فیلد TCP Header Length مشخص می‌کند که سرآیند قطعهٔ TCP چند کلمهٔ ۳۲ بیتی است. از آنجایی که فیلد Options اندازهٔ متغیری دارد فلذا به وجود این فیلد که طول سرآیند را مشخص می‌کند، نیاز خواهد بود. از دیدگاه فنی این فیلد «نقطهٔ شروع داده‌ها در هر قطعه» را بر مبنای کلمات ۳۲ بیتی مشخص می‌کند ولی از دیدگاه دیگر می‌توان مقدار این فیلد را «طول سرآیند قطعهٔ TCP» بر مبنای کلمه فرض کرد ولیکن تأثیر هر دوی این تعابیر یکی است.

در ادامه یک فیلد شش بیتی بلااستفاده آمده است. این حقیقت که از چنین فیلدی برای حدود ربع قرن استفاده نشده، مؤید آن است که TCP با تفکر و بینش بسیار جامعی طراحی شده است. کمتر پروتکلی برای اصلاح اشکالات طرح اصلی TCP، از این فیلد استفاده کرده است.

در ادامه شش پرچم تک‌بیتی (Flag) آمده است: اگر درون فیلد Urgent Pointer مقدار معتبری قرار گرفته باشد، بیت پرچم URG به ۱ تنظیم می‌شود. مقدار فیلد Urgent Pointer مشخص می‌کند که «داده‌های اضطراری» (نسبت به اولین بایت داده‌های قطعهٔ جاری) از چه موقعیتی شروع می‌شوند. در حقیقت این فیلد نقش «پیام وقفه» (Interrupt Message) را ایفاء می‌کند. همانگونه که قبلاً اشاره کردیم این فیلد روش سرراستی برای سگنال

---

۱. به عبارتی اگر در فیلد شمارهٔ تصدیق مثلاً عدد ۱۲۳۴۵ درج شده باشد بدین معنا تعبیر می‌شود که تا بایت ۱۲۳۴۴ دریافت شده و باید از بایت شمارهٔ ۱۲۳۴۵ به بعد ارسال شود. –م

دادن فرستنده به گیرنده (از راه دور) است.

برای آن که مشخص شود فیلد «شمارهٔ تصدیق» (Acknowledgement Number) دارای مقدار معتبر است، بیت پرچم ACK به ۱ مقداردهی می‌شود. اگر بیت ACK معادل صفر باشد، قطعهٔ TCP جاری دارای هیچ «شمارهٔ تصدیق» معتبری نیست و مقدار آن نادیده گرفته می‌شود.

بیت PSH مشخص کننده آن است که داده‌های قطعهٔ جاری باید سریعاً تحویل داده شود (PUSH شود). با تنظیم این بیت در سرآیند قطعهٔ TCP ، از گیرنده خواهش می‌شود که داده‌های موجود در قطعه را بلافاصله تحویل برنامهٔ کاربردی بدهد و تا پر شدن بافر (که معمولاً برای افزایش کارآیی و سرعت کاربرد دارد) منتظر نماند.

از بیت RST زمانی استفاده می‌شود که TCP به هر دلیلی (مثل از کار افتادن ماشین میزبان) بخواهد یک اتصال را بطور ناگهانی و یکطرفه قطع کند. همچنین زمانی که TCP بخواهد یک قطعهٔ نامعتبر یا تلاش برای برقراری یک اتصال را نپذیرد از این بیت استفاده می‌کند. کلاً هر گاه قطعه‌ای دریافت شود که در آن بیت RST به ۱ تنظیم شده، نشان‌دهندهٔ وجود یک مشکل است.

از بیت SYN برای برقراری یک اتصال استفاده می‌شود. در حقیقت تقاضای برقراری اتصال با ارسال قطعه‌ای با مشخصات SYN=1 و ACK=0 انجام می‌گیرد که در آن SYN=1 علامت تقاضای برقراری ارتباط و ACK=0 به معنای عدم وجود مقدار معتبر در فیلد Achnowledgement Number است. پاسخ به این تقاضا به شکل SYN=1 و ACK=1 است و در فیلد Ack.No مقدار معتبری وجود دارد. [بعداً بدین مورد خواهیم پرداخت.] در اصل، بیت SYN برای اعلام هر یک از پیامهای CONNECTION REQUEST و CONNECTION ACCEPTED بکار می‌رود و تمایز بین این دو پیام با بیت ACK مشخص می‌شود.

بیت FIN برای خاتمه دادن به یک اتصال بکار می‌رود. ۱ بودن این بیت بدان معناست که فرستنده، دادهٔ دیگری برای ارسال ندارد. ولیکن پروسه‌ای که با ارسال چنین قطعه‌ای اتصال را از سمت خود می‌بندد می‌تواند تا هر زمانی به دریافت داده‌های طرف مقابل ادامه بدهد. قطعات حاوی بیت SYN=1 یا FIN=1 دارای شمارهٔ ترتیب هستند، فلذا تضمین می‌شود که این بسته‌ها به درستی پردازش خواهند شد.[۱]

کنترل جریان در TCP به کمک یک پنجرهٔ لغزان با اندازه متغیر انجام می‌شود. مقدار درج شده در فیلد Window Size در هر بستهٔ TCP، به طرف مقابل تفهیم می‌کند که از بایتی که شمارهٔ آن در فیلد Acknowledgement Number درج شده، به اندازهٔ چند بایت حق ارسال داده دارد. درج عدد صفر در فیلد Window Size کاملاً قانونی و معتبر است و در حقیقت بیان می‌کند که اگرچه تا بایت شمارهٔ Acknowledgement Number-1 ، دریافت شده ولیکن به دلیل کمبود شدید فضای بافر، تا اطلاع ثانوی نمی‌تواند پذیرای دادهٔ بیشتری باشد. گیرنده بعداً می‌تواند به فرستنده مجوز ارسال بدهد: این کار با فرستادن یک بسته که در آن فیلد Acknowledgement Number همان مقدار قبلی را دارد و فیلد Window Size آن غیرصفر است، صورت می‌گیرد.

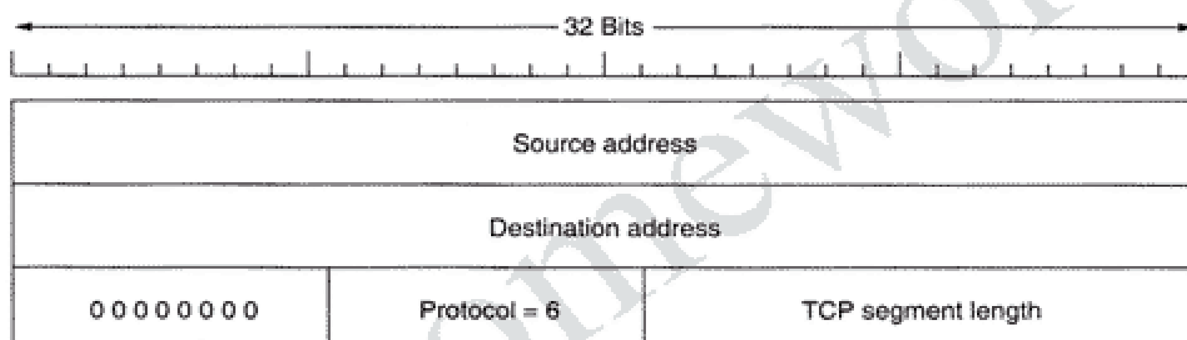
در پروتکلهای فصل سوم، تصدیق وصول فریمها و مجوز ارسال فریمهای جدید یکی هستند. [یعنی دریافت یک پیغام Ack، ضمن تصدیق وصول بسته به فرستنده اجازه می‌دهد که بستهٔ جدیدی بفرستد.] این مسئله از آنجا ناشی می‌شود که طول پنجرهٔ هر یک از پروتکلهای یاد شده ثابت و بدون تغییر است. در پروتکل TCP، اعلام وصول داده‌ها (یعنی Ack) و مجوز ارسال دادهٔ بیشتر از هم تفکیک شده‌اند. در نتیجه یک گیرنده می‌تواند بگوید:

---

۱. به عبارت دیگر اگر به هر دلیلی یک بستهٔ FIN=1 برای پروسه‌ای ارسال شود، پردازش نخواهد شد مگر آن که شمارهٔ ترتیب آن صحیح و دقیق باشد.

«من تا بایت شمارهٔ k را به درستی دریافت کرده‌ام ولیکن فعلاً تمایلی‌به دریافت دادهٔ دیگر ندارم!» تفکیک این دو مفهوم و داشتن پنجره‌ای با طول متغیر، قابلیت انعطاف بیشتری به پروتکل اعطا می‌کند. در ادامه این موضوع را به تفصیل بررسی خواهیم کرد.

فیلد Checksum یک کد کشف خطا است و جهت اطمینان از صحت داده‌ها کاربرد دارد. این کد حاصل جمع سرآیند، داده‌ها و یک «شبه‌سرآیند فرضی» (Pseudoheader) است. قالب شبه‌سرآیند فرضی در شکل ۶-۳۰ مشخص شده است. برای محاسبهٔ این کُد ابتدا فیلد Checksum صفر فرض می‌شود و در صورت فرد بودن تعداد بایتها، تعدادی صفر زائد به انتهای داده‌ها اضافه می‌گردد تا تعداد بایتها زوج شود. الگوریتم محاسبهٔ Checksum بسیار ساده است: مجموعهٔ بایتها به صورت کلمات ۱۶ بیتی (یعنی دو بایت دو بایت) با هم جمع شده و حاصل جمع به صورت «متمم ۱» (One's Complement) منفی می‌شود و درون فیلد Checksum قرار می‌گیرد. نتیجتاً وقتی در گیرنده این محاسبه بر روی کل قطعه (شامل فیلد Checksum) انجام می‌شود نتیجهٔ آن باید صفر باشد. در غیر این صورت داده‌ها قابل اعتماد و سالم نیستند.



شکل ۶-۳۰. شبه‌سرآیندی که در محاسبهٔ کد کشف خطای TCP Checksum دخالت داده می‌شود.

شبه‌سرآیند (Pseudoheader) از فیلدهای زیر تشکیل شده‌اند: آدرسهای IP سی و دو بیتی مبداء و مقصد، شمارهٔ پروتکل TCP (یعنی عدد ۶) و تعداد کل بایتهای قطعهٔ TCP (شامل سرآیند آن). وارد کردن این شبه‌سرآیند در محاسبهٔ مقدار Checksum اگرچه به کشف بسته‌هایی که به اشتباه دریافت شده‌اند کمک خواهد کرد ولیکن اصول و قواعد تفکیک سلسله‌مراتب پروتکلها و مخفی ماندن جزئیات هر لایه را نقض می‌کند چراکه آدرسهای IP متعلق به لایهٔ IP است و ربطی به لایهٔ TCP ندارد. UDP نیز برای محاسبهٔ کد Checksum از همین شبه سرآیند استفاده می‌کند.

فیلد Options برای درج گزینه‌های اختیاری در قطعهٔ TCP تعریف شده است و می‌توان از آن برای اضافه کردن فیلدهایی که در پروتکل TCP پیش‌بینی نشده، استفاده کرد. مهمترین گزینهٔ اختیاری، گزینه‌ای است که به ماشین میزبان اجازه می‌دهد طول حداکثر قطعات TCP که تمایل به پذیرش آنها را دارد، به اطلاع طرف مقابل برساند. استفاده از قطعات بزرگ، کارآمدتر از ارسال قطعات کوچک است زیرا سربار ناشی از سرآیند ۲۰ بایتی، بر روی دادهٔ بیشتری سرشکن می‌شود ولی ممکن است ماشینهای میزبان کوچک از عهدهٔ پذیرش قطعات بزرگ برنیایند. در خلال برقراری یک اتصال، هر یک از طرفین طول حداکثر قطعات مورد پذیرش خود را به اطلاع طرف مقابل خود می‌رساند. اگر ماشینی از این گزینه استفاده نکند، اندازهٔ پیش فرض داده‌ها در هر قطعه ۵۳۶ بایت خواهد بود. تمام ماشینهای میزبان در اینترنت موظف به پذیرش قطعات TCP با اندازهٔ ۵۵۶=۲۰+۵۳۶ بایت هستند. الزامی به یکسان بودن اندازهٔ قطعات ارسالی در دو جهت وجود ندارد.

برای خطوط با پهنای باند بالا یا تأخیر زیاد (یا هر دو)، پنجرهٔ ۶۴کیلوبایتی می‌تواند مشکل‌زا باشد.[1] در یک خط T3 (با سرعت 44.736Mbps)، تخلیه و ارسال یک بافر ۶۴کیلوبایتی فقط ۱۲ میلی‌ثانیه طول می‌کشد. یا مثلاً اگر تأخیر انتشار رفت‌وبرگشت [یعنی رفت بسته و برگشت Ack آن] ۵۰میلی‌ثانیه طول بکشد (که برای خطوط فیبرنوری بین قاره‌ای کاملاً طبیعی است)، فرستنده سه چهارم از زمان مفید خود را در انتظار بازگشت پیام اعلام وصول (Ack) تلف می‌کند. برای ارتباطات ماهواره‌ای وضع از این هم بدتر است.[2] استفاده از پنجره‌ای با طول بزرگتر از ۶۴کیلوبایت می‌تواند از توقف فرستنده به دلیل پر شدن فضای پنجره جلوگیری کند ولیکن با فیلد ۱۶ بیتی Window Size نمی‌توان اندازهٔ چنین بافری را اعلام کرد. در RFC 1323 یک گزینهٔ اختیاری به نام Window Scale پیشنهاد شده که به کمک آن فرستنده و گیرنده می‌توانند «**ضریب مقیاس پنجره**» را به اطلاع یکدیگر برسانند. این گزینهٔ اختیاری در فیلد Options قرار خواهد گرفت و اجازه می‌دهد که هر یک از طرفین بتوانند فیلد Window Size خود را تا ۱۴ بیت به سمت چپ شیفت بدهند. بدین ترتیب اندازهٔ پنجره می‌تواند تا 2^{30} بایت (یک گیگابایت) افزایش یابد. امروزه در اغلب پیاده‌سازیهای TCP، از این گزینه حمایت می‌شود.

گزینهٔ دیگری که در RFC 1106 پیشنهاد شده و در اغلب پیاده‌سازیهای TCP از آن پشتیبانی می‌شود آن است که به جای استفاده از پروتکل Go Back n (عقبگرد به اندازهٔ n) از پروتکل «**تکرار انتخابی**» (Selective Repeat) بهره گرفته شود. اگر گیرنده، ابتدا قطعه‌ای خراب و به دنبال آن چندین قطعهٔ بزرگ و سالم دریافت نماید، در صورتی که از پروتکل معمولی TCP استفاده شود، فرستنده پس از انقضای مهلت مقرر تمام قطعات اعلام وصول نشده را از نو ارسال خواهد کرد (حتی بسته‌هایی را که سالم رسیده‌اند) زیرا از پروتکل Go Back n بهره گرفته شده است. در RFC 1106 مفهوم جدیدی به نام NAK (پیغام عدم وصول) معرفی شده که به گیرنده اجازه می‌دهد قطعهٔ خاصی را درخواست بدهد. پس از دریافت قطعهٔ ناقص، وصول تمام داده‌هایی که از قبل بافر شده‌اند به یکباره تصدیق می‌شود و بدین ترتیب حجم داده‌هایی که بیهوده ارسال مجدد می‌شوند، کاهش خواهد یافت.

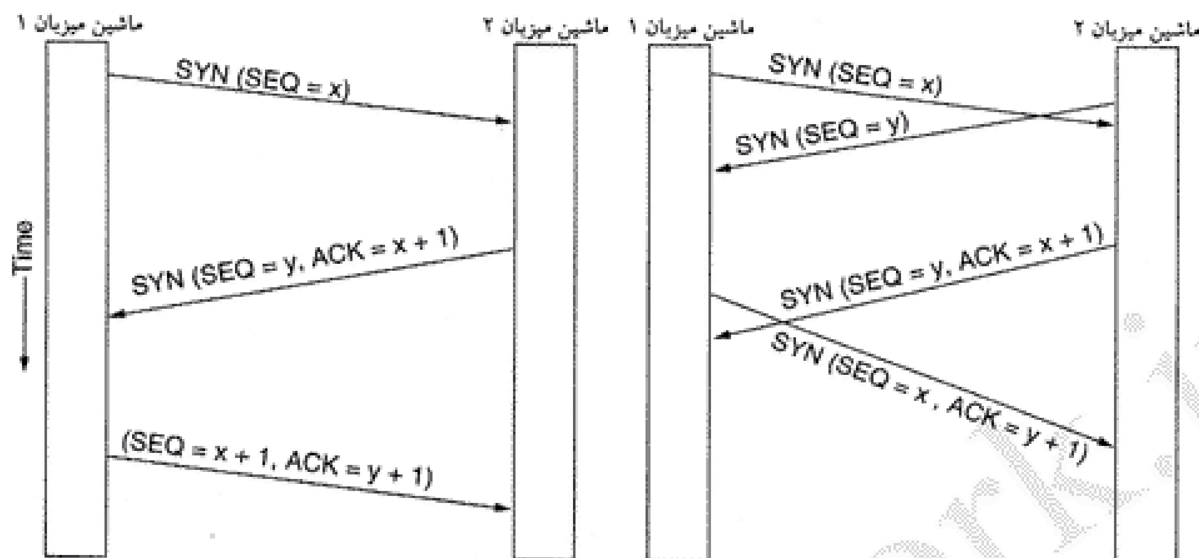
۶-۵-۵ برقراری اتصال TCP

در پروتکل TCP برای برقراری اتصال از روش «**دست‌تکانی سه‌مرحله‌ای**» (Three-Way Handshake) (که در بخش ۶-۲-۲ تشریح شد) بهره گرفته می‌شود. برای آن که اتصالی ایجاد شود باید یکی از طرفین (که آن را سرویس‌دهنده می‌نامیم) از طریق اجرای توابع اولیهٔ LISTEN و ACCEPT منتظر تقاضاهای ورودی باقی بماند. (البته می‌تواند مبداء مورد نظر خود را تعیین کند یا آن که مبداء خاصی را مدّ نظر نداشته باشد.) سمت مقابل (یعنی مشتری)، تابع اولیه CONNECT را اجرا کرده و آدرس IP و پورت کسی را که می‌خواهد با آن اتصال برقرار کند و همچنین طول حداکثر قطعات TCP مورد پذیرش (و در صورت نیاز برخی از اطلاعات کاربری نظیر کلمهٔ عبور) را مشخص می‌نماید. تابع اولیهٔ CONNECT، یک قطعهٔ TCP با مشخصات SYN=1 و ACK=0 برای طرف مقابل فرستاده و منتظر برگشت پاسخ باقی می‌ماند.

وقتی چنین بسته‌ای به مقصد می‌رسد، واحد TCP در آنجا ابتدا بررسی می‌کند که آیا پروسه‌ای توسط تابع LISTEN در حال گوش دادن به شمارهٔ پورت مشخص شده در فیلد شمارهٔ پورت مقصد (Destination Port) هست یا خیر؟ اگر چنین نبود با ارسال یک قطعهٔ TCP حاوی بیت RST=1، تقاضای برقراری اتصال را ردّ می‌کند.

اگر پروسه‌ای در حال گوش دادن به پورت مربوطه باشد، قطعهٔ TCP ورودی، بدان پروسه تحویل داده خواهد

۱. از آنجایی که فیلد Window Size شانزده بیتی است لذا طول پنجره به 64KB (۶۵۵۳۵بایت) محدود شده‌است. -م
۲. تأخیر رفت بسته و برگشت Ack آن، در کانالهای ماهواره‌ای ژئوسنکرون حدود ۸۰۰میلی‌ثانیه است. -م



شکل ۶-۳۱. (الف) ایجاد یک اتصال TCP در شرایط معمولی (ب) تلاقی دو تماس.

شد. پروسهٔ مربوطه می‌تواند آن اتصال را بپذیرد یا رد کند. اگر پذیرفته شود یک قطعهٔ تصدیق (Acknowledgement) برگردانده خواهد شد. توالی ارسال قطعات TCP جهت برقراری اتصال، در شکل ۶-۳۱-الف (در حالت طبیعی) نشان داده شده است.

در حالتی که تصادفاً دو ماشین بطور همزمان بخواهند اتصالی بین دو سوکت یکدیگر برقرار نمایند، رخدادهایی با ترتیب شکل ۶-۳۱-ب اتفاق می‌افتد. نتیجهٔ نهایی این رخدادها آن است که فقط و فقط یک اتصال برقرار می‌شود نه دو تا، زیرا هویت هر اتصال برحسب نقاط پایانی آن [یعنی آدرسهای IP و پورت دو پروسهٔ پایانی] شناسایی می‌شود؛ اگر اولین اتصال برقرار شده با شناسهٔ (x,y) مشخص شود، اتصال دوم نیز دارای همین شناسه خواهد بود و در جدول اتصالات فعال فقط یک درایه (Entry) با شناسهٔ (x,y) درج خواهد شد.

شمارهٔ ترتیب اولیهٔ پیشنهادی در هر اتصال به دلیلی که قبلاً بدان اشاره کردیم از صفر شروع نخواهد شد. برای انتخاب این شماره از ساعتی استفاده می‌شود که هر ۴ میکروثانیه تیک می‌زند. برای اطمینان بیشتر هر گاه ماشینی از کار بیفتد می‌تواند به مدت حداکثر طول عمر بسته‌هایی که از اتصال قبلی که در جایی از شبکهٔ اینترنت سرگردان مانده‌اند، صبر کند و سپس از نو راه‌اندازی شود.

۶-۵-۶ خاتمه دادن به اتصال TCP

هر چند اتصالات TCP، دو طرفهٔ کامل (Full Duplex) هستند ولی برای فهم چگونگی قطع اتصال، بهتر است یک اتصال TCP را به صورت یک جفت اتصال یکطرفه (Simplex) در نظر بگیریم. هر اتصال یکطرفه، مستقل از جفت خود قطع می‌شود. برای خاتمه دادن به یک اتصال، هر یک از طرفین می‌توانند یک قطعهٔ TCP که در آن بیت FIN به ۱ تنظیم شده، ارسال نمایند. چنین قطعه‌ای بدین معناست که دادهٔ دیگری جهت ارسال وجود ندارد. وقتی دریافت این قطعه تصدیق شد اتصال در یکی از جهات، خاتمه می‌یابد ولیکن جریان داده‌ها در جهت مخالف می‌تواند بطور نامحدود ادامه داشته باشد. وقتی که اتصال در هر دو جهت قطع شد، اتصال TCP خاتمه می‌یابد. طبیعتاً برای خاتمه دادن به یک اتصال، به ارسال چهار قطعهٔ TCP نیاز است: یکی برای FIN و یکی برای ACK، در هر یک از جهات. با این حال ممکن است اولین ACK برگشتی با FIN طرف مقابل در یک بسته ادغام شود و تعداد این بسته‌ها به سه تا کاهش یابد.

دقیقاً مشابه با تماسهای تلفنی که در آن هر دو نفر می‌توانند با خداحافظی، گوشی تلفن خود را بگذارند، دو پروسهٔ انتهایی یک اتصال نیز ممکن است بطور همزمان قطعه‌ای حامل FIN=1 بفرستند. هر یک از این تقاضاهای قطع اتصال به روش معمولی تصدیق شده و به اتصال خاتمه داده می‌شود.

برای آن که مشکل «دو سپاه» (بخش ۶ـ۲ـ۳) پیش نیاید، از تایمر نیز استفاده شده است. اگر پاسخ به FIN به مدت دو برابر طول عمر حداکثر هر بسته دریافت نشود، فرستندهٔ FIN، بطور یک جانبه به اتصال خاتمه خواهد داد. طرف مقابل نیز عاقبت متوجه خواهد شد که هیچکسی در طرف مقابل به او گوش نمی‌دهد و مهلت او نیز منقضی می‌شود و به اتصال خاتمه خواهد داد. اگرچه این راهکار چندان کامل و مطلوب نیست ولیکن از آنجایی که از لحاظ تئوری هیچ راهکار مطمئن و کاملی وجود ندارد مجبور به برگزیدن این راهکار هستیم. البته این مشکلات در عمل به ندرت بروز می‌کند.

۶ـ۵ـ۷ مدلسازی فرآیند مدیریت اتصال در TCP

مراحل لازم برای برقراری و ختم اتصال در TCP را می‌توان در قالب یک **«ماشین حالت محدود»** (Finite State Machine) با یازده وضعیت مختلف که در جدول ۶-۳۲ فهرست شده، نشان داد. در هر **«وضعیت»** وقوع برخی از **«رخدادها»** معتبر هستند. وقتی یک رخداد معتبر حادث می‌شود سلسله‌ای از **«کنشها»** (Actions) انجام خواهد شد. برای برخی دیگر از رخدادها نیز، فقط به گزارش خطا بسنده می‌شود.

هر اتصال از وضعیت CLOSED آغاز می‌شود. اگر یکی از رخدادهای «بازکردن غیرفعال» (با اجرای تابع اولیهٔ LISTEN) یا «بازکردن فعال»[1] (با اجرای تابع اولیهٔ CONNECT) رخ بدهد، اتصال از این «وضعیت» خارج خواهد شد. اگر طرف مقابل اتصال دقیقاً در وضعیت معکوس قرار داشته باشد یک اتصال برقرار شده و وضعیت جدید اتصال، ESTABLISHED خواهد شد. قطع یک اتصال می‌تواند توسط هر یک از طرفین شروع شود. وقتی فرآیند قطع اتصال تکمیل شد، اتصال به وضعیت قبلی CLOSED باز خواهد گشت.

وضعیت (حالت)	توصیف
CLOSED	هیچ اتصالی فعال یا معلق (و منتظر) نیست.
LISTEN	سرویس‌دهنده منتظر یک تماس ورودی (تقاضای اتصال) است.
SYN RCVD	یک تقاضای برقراری اتصال دریافت شده است. منتظر ACK آن است.
SYN SENT	برنامه کاربردی (مشتری) شروع به ایجاد یک اتصال کرده است.
ESTABLISHED	وضعیت عادی مبادله داده (وضعیت برقراری اتصال)
FIN WAIT 1	برنامه کاربردی اعلام کرده کارش به اتمام رسیده است.
FIN WAIT 2	طرف مقابل نیز با قطع اتصال و خاتمه تماس موافقت کرده است.
TIMED WAIT	حالت انتظار برای آنکه تمام بسته‌های سرگردان از بین بروند.
CLOSING	طرفین بطور همزمان سعی در بستن اتصال کرده‌اند.
CLOSE WAIT	طرف مقابل مراحل قطع ارتباط را آغاز کرده است.
LAST ACK	حالت انتظار برای آنکه تمام بسته‌های سرگردان ACK از بین بروند.

شکل ۶-۳۲. وضعیتهایی که در مدل «ماشین حالت محدود» از مدیریت اتصال TCP بکار می‌رود.

۱. Active Open / Passive Open

«ماشین حالت محدود» برای TCP، در شکل ۶-۳۳ به تصویر کشیده شده است. حالتی که در آن ماشین مشتری بطور فعّال به یک سرویس‌دهندهٔ غیرفعال متصل می‌شود با خطوط تیره رنگ نشان داده شده است: خطوط توپر برای تقاضای مشتری و نقطه چین برای پاسخ سرویس‌دهنده. خطوط کمرنگ، توالی رخدادهای نامتعارف را نشان می‌دهند. کنار هر خط در شکل ۶-۳۳ یک زوج مشخصه به صورت event/action (رخداد/کنش) نوشته شده است. هر رخداد می‌تواند ناشی از یک فراخوانی سیستمی باشد که توسط کاربر آغاز می‌شود (مثل فراخوانیهای CONNECT ، LISTEN ، SEND یا CLOSE)، یا در اثر دریافت یک قطعهٔ TCP (مثل SYN ، FIN ، ACK یا RST) بروز کند و یا در اثر انقضای مهلت تایمر حادث شود. «کنشی» که در حین بروز یک رخداد انجام می‌شود ، ارسال یک قطعهٔ کنترلی (مثل SYN ، FIN یا RTS) است یا هیچ کاری صورت نمی‌گیرد (که در شکل با علامت - مشخص شده است). توضیحات لازم درون پرانتز نشان داده شده است.

برای فهم دیاگرام، بهتر آنست که ابتدا مسیر منشعب از مشتری (خطوط توپر ضخیم) و بعد از آن مسیر سرویس‌دهنده (خطوط ضخیم نقطه چین) دنبال شود. وقتی برنامهٔ کاربردی بر روی ماشین مشتری تقاضای CONNECT صادر می‌کند، «واحد TCP» در آن ماشین، ابتدا یک رکورد برای آن اتصال ایجاد کرده و پس از علامتگذاری اتصال در وضعیت SYNSENT یک قطعهٔ SYN ارسال می‌نماید. دقت کنید که ممکن است چندین اتصال از طرف چند برنامهٔ کاربردی بطور همزمان باز شده (یا در حال باز شدن) باشد فلذا به ازای هر اتصال یک «رکورد وضعیت» مستقل ایجاد و وضعیت آن اتصال در رکورد مربوطه درج می‌شود. وقتی قطعهٔ کنترلی حاوی SYN+ACK دریافت می‌شود، TCP با ارسال ACK نهایی، فرآیند دست تکانی سه‌مرحله‌ای را تکمیل کرده و به ESTABLISHED تغییر وضعیت می‌دهد. در این وضعیت می‌توان داده فرستاد یا دریافت کرد.

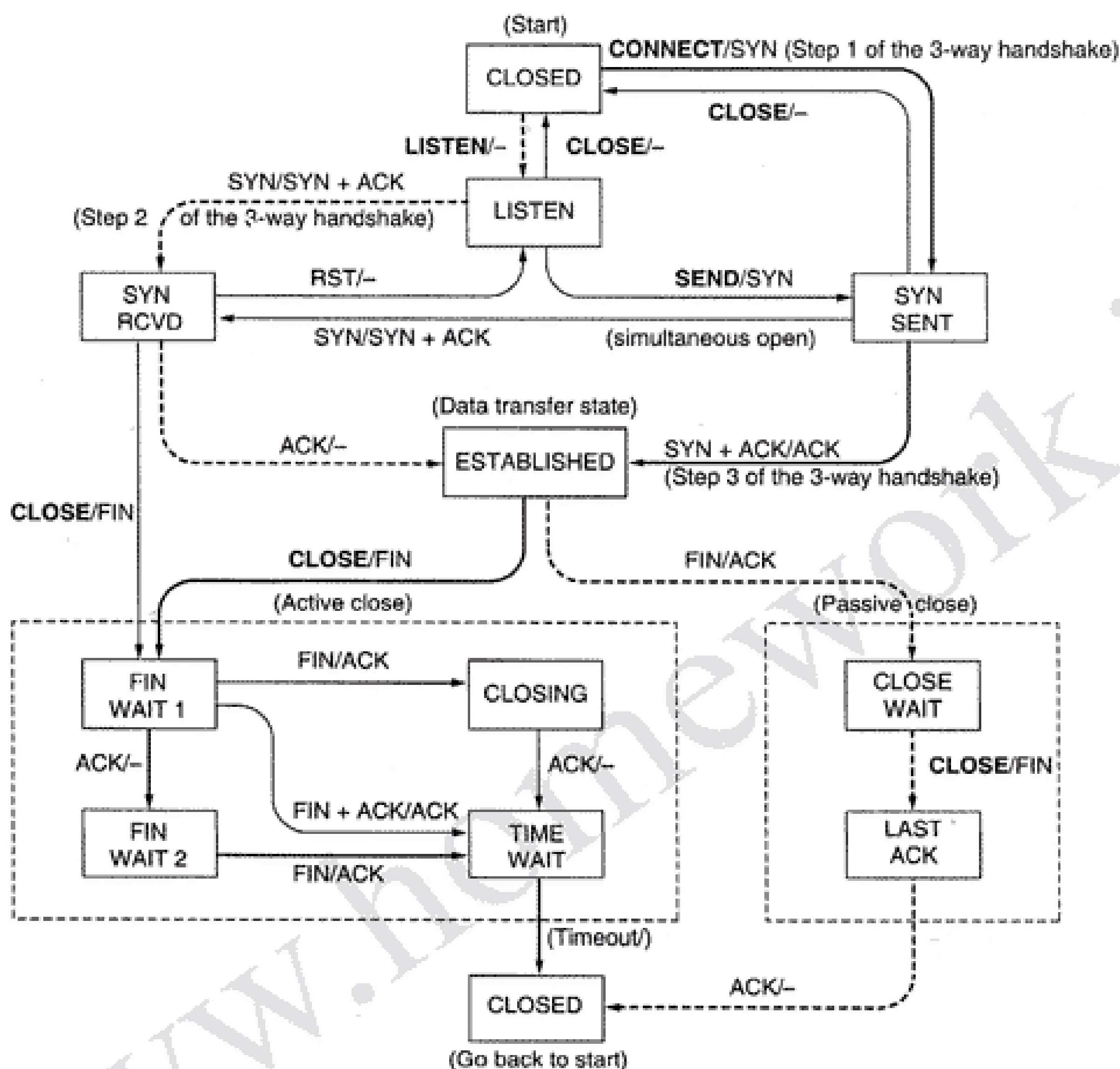
وقتی کار برنامهٔ کاربردی به پایان رسید، تابع اولیهٔ CLOSE را اجرا می‌کند. این کار موجب خواهد شد که «واحد انتقال TCP» یک قطعهٔ کنترلی FIN فرستاده و منتظر ACK مربوطه بماند. (مستطیل نقطه چین با عنوان active close نشان داده شده است.) وقتی ACK مربوطه دریافت شود، وضعیت اتصال به حالت FIN WAIT2 تغییر کرده و یک جهت از اتصال بسته خواهد شد. وقتی طرف مقابل نیز اتصال را ببندد، یک بستهٔ FIN از راه می‌رسد و وصول آن تصدیق می‌شود. حال اگرچه هر دو طرف اتصال را بسته‌اند ولیکن TCP بایستی به اندازهٔ دو برابر طول عمر حداکثر بسته‌ها صبر کند تا تمام بسته‌هایی که احتمالاً از اتصال قبلی در شبکه سرگردان هستند از بین بروند. به محض آن که مهلت چنین تایمری به پایان رسید، TCP رکورد اتصال مربوطه را حذف خواهد کرد.

حال مدیریت اتصال را از دیدگاه سرویس‌دهنده بررسی می‌نماییم: سرویس‌دهنده با انجام عمل LISTEN منتظر می‌ماند تا کسی تماس بگیرد. وقتی یک SYN وارد می‌شود، دریافت آن بلافاصله تصدیق شده و سرویس‌دهنده به وضعیت SYN RCVD وارد می‌شود. وقتی SYN-ACK ارسالی سرویس‌دهنده توسط مشتری اعلام وصول شد، فرآیند دست تکانی سه‌مرحله‌ای تکمیل می‌گردد و سرویس‌دهنده به وضعیت ESTABLISHED می‌رود. انتقال داده‌ها اکنون می‌تواند شروع شود.

وقتی کار مشتری به انجام رسید، اقدام به صدور فرمان CLOSE می‌کند. این کار موجب می‌شود که سرویس‌دهنده، FIN دریافت کند. (مستطیل نقطه چین، خاتمهٔ غیرفعال[1] را نشان می‌دهد.) در این لحظه به سرویس‌دهنده سیگنال داده می‌شود. هر گاه او نیز فرمان CLOSE را صادر کند، یک قطعهٔ حاوی FIN برای مشتری ارسال می‌شود. وقتی مشتری دریافت FIN را تأیید کرد، سرویس‌دهنده به اتصال مربوطه خاتمه داده و رکورد متناظر با آن اتصال را از جدول خود حذف می‌نماید.

---

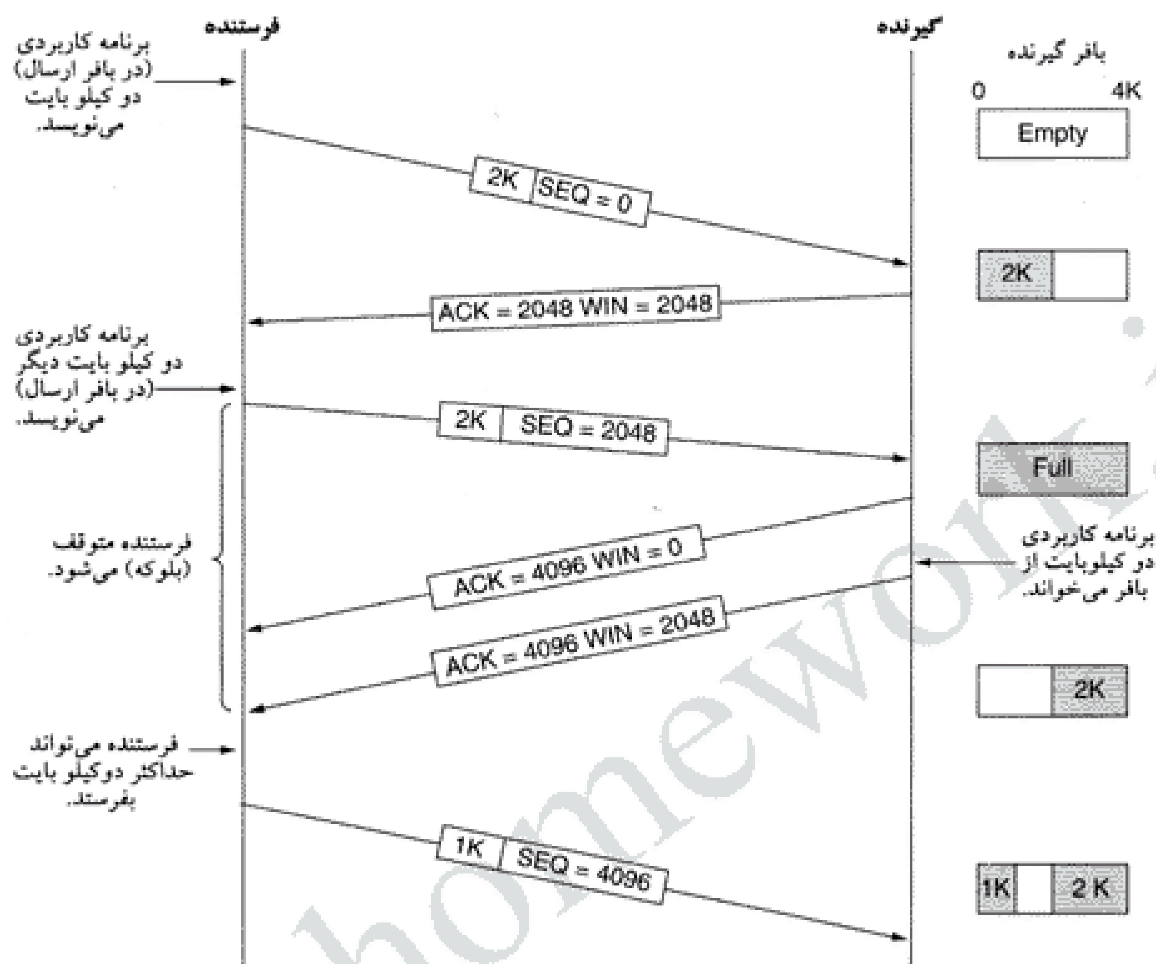
۱. Passive Close



شکل ۶-۳۳. «ماشین حالت محدود» برای مدیریت اتصال TCP. خطوط ضخیم توپر مسیر طبیعی عملکرد مشتری را نشان می‌دهد. خطوط ضخیم نقطه‌چین مسیر طبیعی عملکرد سرویس دهنده را نشان می‌دهد. خطوط نازک توپر بروز رخدادهای نامعمول را نشان می‌دهد. بر روی هر «گذار» (Transition) برچسبی وجود دارد که «رخداد» و «کنش» مربوطه را مشخص می‌کند.

۶-۵-۸ سیاستهای انتقال در TCP

همانگونه که قبلاً اشاره شد، مدیریت پنجره در TCP، وابستگی مستقیمی به تصدیق دریافت داده‌ها (Ack) ندارد (در حالی که در اغلب پروتکلهای لایهٔ پیوند داده اینگونه است). به عنوان مثال در شکل ۶-۳۴ فرض کنید که گیرنده ۴۰۹۶ بایت فضای بافر در اختیار دارد. اگر فرستنده یک قطعهٔ ۲۰۴۸ بایتی ارسال کند و به درستی و بدون خطا دریافت شود، گیرنده بلافاصله وصول آنرا تأیید خواهد کرد. با این وجود، از آنجایی که پس از دریافت این قطعه، فقط ۲۰۴۸ بایت از فضای بافر خالی است (مگر آن که برنامهٔ کاربردی بخشی از داده‌ها را از بافر بردارد) فلذا به طرف مقابل خود اعلام می‌دارد که از بایت بعدی (که شمارهٔ آن در فیلد Ack.No. مشخص شده) فقط حق ارسال ۲۰۴۸ بایت داده دارد. یا بعبارت فنی اعلام می‌کند، پنجرهٔ ۲۰۴۸ بایتی از شمارهٔ بایتی که در فیلد ۳۲ بیتی



شکل ۶-۳۴. مدیریت پنجره در TCP.

Ack. No. مشخص شده، آغاز می‌گردد.

حال فرستنده، ۲۰۴۸ بایت دادهٔ دیگر ارسال می‌نماید و دریافت آنها نیز تأیید می‌شود ولیکن، اندازهٔ پنجره، صفر اعلام می‌شود. فرستنده باید متوقف شود و آنقدر منتظر بماند تا پروسهٔ کاربردی گیرندهٔ داده‌ها، مقداری داده از بافر بردارد و TCP بتواند پنجرهٔ بزرگتری را اعلام کند.

وقتی اندازهٔ پنجره صفر اعلام شده، فرستنده عموماً نمی‌تواند قطعهٔ دیگری ارسال کند مگر در دو مورد استثنا: اول «داده‌های اضطراری» (Urgent Data)؛ برای آنکه به کاربر اجازه بدهیم مثلاً پروسهٔ کاربردی اجرا شده بر روی ماشین راه دور را از بین ببرد.[1] دوم آن که فرستنده ممکن است قطعه‌ای حاوی یک بایت داده ارسال کند تا گیرنده وادار شود اندازهٔ پنجرهٔ خود و شمارهٔ ترتیب بایتی را که از آن به بعد منتظر دریافت است، از نو اعلام نماید. استاندارد TCP، از این گزینه برای اجتناب از بروز بن‌بست بهره گرفته است تا در صورت عدم اعلام طول پنجره، پروسهٔ متوقف شده در سمت گیرنده، تا ابد منتظر نماند.

فرستنده ملزم نیست که به محض تحویل گرفتن داده‌ها از برنامهٔ کاربردی، فوراً آنها را ارسال نماید. گیرندهٔ داده‌ها نیز مجبور نیست به محض دریافت و در اسرع زمان، دریافت آنها را تصدیق نماید. به عنوان مثال در شکل ۶-۳۴ وقتی اولین دو کیلوبایت از برنامهٔ کاربردی دریافت شد، TCP با اطلاع از آن که ۴ کیلوبایت فضای بافر در

---

۱. به اصطلاح یونیکس killکند.

اختیار دارد، می‌تواند ارسال آن را به تأخیر بیندازد تا ۲ کیلوبایت بعدی نیز دریافت شود و هر چهار کیلوبایت را یکجا ارسال نماید. این آزادی عمل می‌تواند به افزایش کارآیی TCP کمک کند.

حال یک اتصال Telnet را مدّ نظر قرار بدهید که با فشار هر کلید در «ویرایشگر محاوره‌ای» (Interactive Editor)، از راه دور باید واکنش نشان بدهد. در بدترین حالت، وقتی که یک کاراکتر تکی جهت ارسال به «واحد انتقال TCP» تحویل می‌شود، TCP یک قطعهٔ ۲۱ بایتی برای آن تشکیل داده و آن را به IP می‌دهد. IP نیز آن را به صورت یک دیتاگرام ۴۱ بایتی ارسال می‌نماید. (۲۰ بایت سرآیند TCP + ۲۰ بایت سرایند IP + ۱ بایت کاراکتر ارسالی). در سمت گیرنده، TCP بلافاصله وصول آن را با ارسال یک دیتاگرام ۴۰ بایتی تصدیق می‌کند. بعداً وقتی برنامه ویرایشگر این بایت را می‌خواند، TCP بار دیگر مقدار جدید پنجرهٔ خود را که فقط یک واحد افزایش داشته، اعلام می‌دارد. این بسته نیز ۴۰ بایتی است. در آخر نیز وقتی برنامهٔ ویرایشگر، کاراکتر ارسالی را پردازش کرد، آن را در قالب یک بستهٔ ۴۱، بازگشت می‌دهد. (یا به عبارتی آن را Echo می‌کند.) بدین نحو، به ازای هر کاراکتر تایپ شده، ۱۶۴ بایت از پهنای باند مصرف و جمعاً چهار قطعهٔ TCP مبادله می‌شود. وقتی پهنای باند ارزشمند و محدود باشد این روش ، بهیچوجه کارآمد نیست.

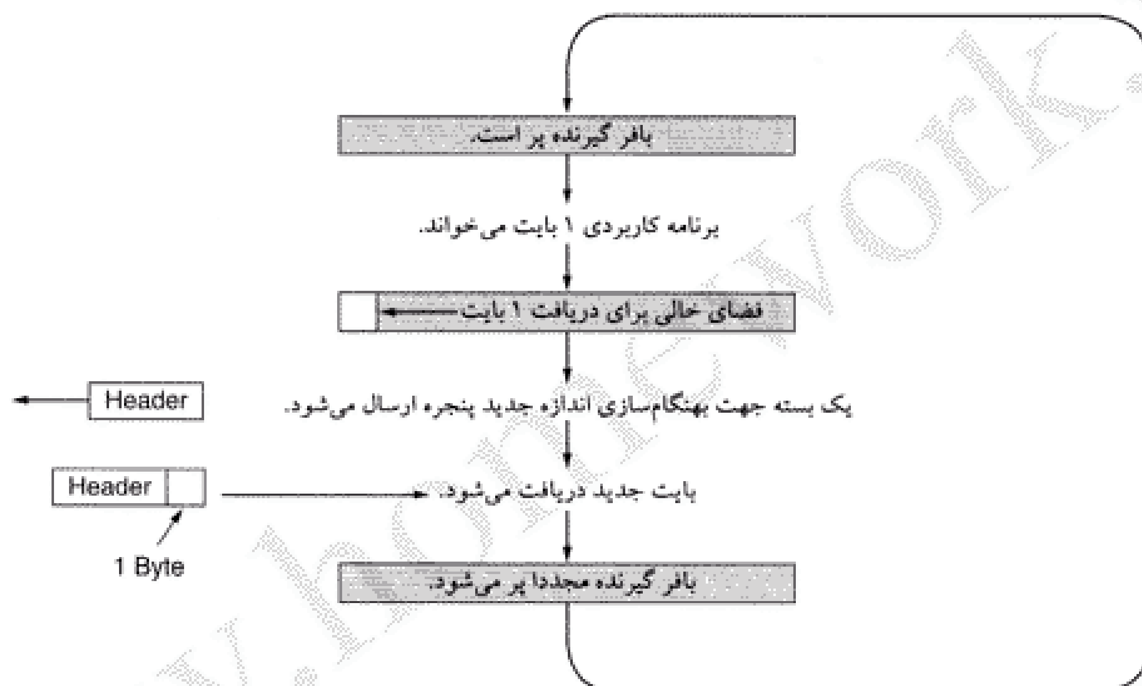
یک راهکار که در بسیاری از پیاده‌سازیهای عملی TCP برای بهینه سازی این وضعیت مورد استفاده قرار گرفته، آنست که اعلام وصول داده‌ها و اعلام اندازهٔ پنجره، به مدت ۵۰۰ میلی‌ثانیه به تعویق بیفتد، بدین امید که داده‌هایی جهت ارسال تولید و در بسته‌ای که باید ارسال می‌شد به رایگان حمل شوند. با فرض آنکه برنامه ویرایشگر در مثال بالا، هر کاراکتر را پس از ۵۰۰میلی‌ثانیه بازگشت بدهد (Echo کند)، فقط به ارسال یک بستهٔ ۴۱ بایتی برای کاربر نیاز است و تعداد بسته‌ها و میزان مصرف پهنای باند به نصف کاهش می‌یابد.

اگرچه این قاعده بار تحمیل شده توسط گیرنده بر روی شبکه را کاهش می‌دهد ولیکن هنوز هم فرستنده بسیار ناکارآمد عمل می‌کند چرا که برای ارسال یک بایت، ۴۱ بایت منتقل می‌شود. یک روش برای کاهش مصرف پهنای باند، **«الگوریتم ناگل»** (Nagle's Algorithm) نام دارد. (Nagle, 1984) آنچه که ناگل پیشنهاد کرد ساده بود: وقتی داده‌ها به صورت تک بایتی جهت ارسال تحویل فرستنده می‌شوند، فرستنده اولین بایت را ارسال می‌نماید بقیهٔ آنها را تا وقتی که دریافت همین یک بایت تأیید شود، بافر می‌کند. پس از اعلام وصول اولین بایت، فرستنده تمام کاراکترهای بافر شده را در قالب یک قطعهٔ TCP به یکباره ارسال می‌کند و مجدداً تا اعلام وصول آن، به بافر کردن کاراکترها ادامه می‌دهد. با این روش، اگر سرعت تایپ کاربر زیاد و شبکه بسیار کُند باشد، تعداد قابل توجهی کاراکتر در قالب یک قطعهٔ TCP ارسال می‌شود و پهنای باند مورد نیاز، کاهشی چشمگیر خواهد داشت. مضاف بر این، «الگوریتم ناگل» اجازه می‌دهد که بستهٔ جدید فقط زمانی ارسال شود که داده‌ها نیمی از فضای پنجره را پر کرده باشند و یا از طول مجاز یک قطعهٔ TCP بیشتر شده باشد.

«الگوریتم ناگل» بطور گسترده‌ای در پیاده‌سازیهای عملی TCP بکار گرفته شده است ولیکن در برخی از مواقع، غیرفعال کردن آن مفیدتر است. خصوصاً وقتی که از برنامهٔ کاربردی X Window در اینترنت استفاده می‌شود، حرکت ماوس باید به کامپیوتر راه دور ارسال شود. (سیستم X Window یک سیستم گرافیکی مبتنی بر پنجره است که در اغلب سیستمهای یونیکس از آن استفاده می‌شود. کاربران راه دور می‌توانند از آن جهت ورود از راه دور به سیستم یونیکس و تعامل با آن بهره بگیرند.) استفاده از الگورتیم ناگل موجب می‌شود که TCP ارسال حرکات ماوس را جمع‌آوری کرده تا آنها را به یکباره ارسال نماید و این کار حرکت ماوس را غیرطبیعی جلوه می‌دهد و اسباب ناخشنودی کاربران را فراهم می‌آورد.

یکی دیگر از مشکلاتی که می‌تواند کارآیی TCP را کاهش بدهد، **«سندرم پنجرهٔ ناموزون»** نام دارد. (Silly Window Syndrome, Clark, 1982) این مسئله زمانی رخ می‌دهد که داده‌ها در قالب بلوکهای بزرگ به

واحد انتقال TCP تحویل شود ولیکن برنامهٔ کاربردی گیرنده در طرف مقابل ، داده‌ها را به صورت بایت به بایت بخواند! برای درک بهتر این مشکل به شکل ۶-۳۵ نگاه کنید. در ابتدا بافر TCP در سمت گیرنده پر است و فرستنده از این موضوع اطلاع دارد. (به عبارت دیگر اندازهٔ پنجره صفر اعلام شده است.) سپس برنامهٔ کاربردی از استریم TCP (یا به عبارتی از بافر)، یک کاراکتر می‌خواند. این کار موجب می‌شود که TCP با ارسال بسته‌ای مقدار جدید پنجرهٔ خود را به اطلاع فرستنده برساند و به او تفهیم کند که اجازهٔ ارسال فقط یک بایت دارد! فرستنده اطاعت کرده و یک بایت ارسال می‌کند. حال بافر مجدداً پر می‌شود و ضمن تصدیق وصول این یک بایت، اندازهٔ پنجره صفر اعلام می‌شود. این رفتار می‌تواند تا بی‌نهایت ادامه داشته باشد.



شکل ۶-۳۵. سندروم پنجرهٔ ناموزون (Silly Window Syndrome).

راه حل پیشنهادی «کلارک» (Clark) آن بود که گیرنده برای یک بایت، مقدار جدید اندازهٔ پنجرهٔ خود را اعلام ننماید؛ در عوض باید آنقدر منتظر شود تا فضای موجود در بافر به حد متناسبی برسد، آنگاه این مقدار اعلام گردد. به ویژه، گیرنده نباید مقدار جدید اندازهٔ پنجرهٔ خود را اعلام کند مگر آن که قادر باشد یک قطعه داده را با طولی که در ابتدای برقراری اتصال به پذیرش آن متعهد شده، در بافر خود بپذیرد یا آن که حداقل نیمی از بافرش خالی شده باشد.

مضاف بر این، فرستنده می‌تواند با نفرستادن قطعات کوچک، به کاهش مشکل کمک کند: فرستنده باید سعی کند که قبل از ارسال یک قطعه آنقدر منتظر بماند تا داده‌های کافی متناسب با اندازهٔ یک قطعهٔ کامل یا معادل با نصف بافر گیرنده جمع‌آوری شود. (اندازهٔ بافر گیرنده را می‌توان براساس اعلامهای متوالی اندازهٔ پنجره، تخمین زد.)

الگوریتم ناگل و راه حل کلارک برای درمان «سندرم پنجرهٔ ناموزون»، مکمل یکدیگر هستند. ناگل سعی می‌کرد مشکلی را حل کند که در اثر تحویل بایت به بایت داده‌ها توسط برنامهٔ کاربردی به TCP پدید می‌آید. کلارک نیز سعی می‌کرد عکس این مشکل را یعنی وقتی که برنامهٔ کاربردی داده‌های خود را از TCP بایت به بایت

تحویل می‌گیرد، حل و فصل نماید. هر دوی این راهکارها معتبرند و در کنار هم کار می‌کنند. هدف آن است که فرستنده، قطعات کوچک نفرستد و گیرنده نیز قطعات را در اندازهٔ کوچک تحویل نگیرد.

در سمت گیرنده، TCP می‌تواند به غیر از بکارگیری روش کلارک، به گونهٔ دیگری نیز برای بهبود کارآیی اقدام نماید. همانند فرستنده، گیرنده نیز می‌تواند داده‌ها را بافر نماید؛ یعنی درخواست برنامهٔ کاربردی جهت خواندن داده‌ها را آنقدر معلق نگاه دارد تا آن که یک تودهٔ بزرگ داده جمع‌آوری شود. انجام این کار تعداد فراخوانیهای TCP را کاهش داده و طبعاً سربار سیستم کمتر می‌شود. البته این کار زمان تایمر و تأخیر اجرای فرمان READ را افزایش خواهد داد ولیکن برای کاربردهایی نظیر انتقال فایل، کارآیی بیشتر ارجح‌تر از زمان پاسخ سریع است.

مورد دیگری که گیرنده با آن روبروست، دریافت قطعات دادهٔ نامرتب است. گیرنده می‌تواند بر حسب شرایط بسته‌هایی را که خارج از ترتیب می‌رسند، نگاه دارد یا آنها را دور بریزد. البته وصول داده‌ها را می‌توان فقط زمانی اعلام کرد که همهٔ داده‌ها تا شماره‌ای که اعلام می‌شود دریافت شده باشد. اگر گیرنده قطعات ۰، ۱، ۲، ۴، ۵، ۶ و ۷ را (به استثنای ۳) دریافت کند تنها می‌تواند دریافت داده‌ها تا آخرین بایت قطعهٔ شمارهٔ ۲ را تصدیق نماید. وقتی مهلت فرستنده منقضی می‌شود، تمام قطعات از شمارهٔ ۳ به بعد از نو ارسال می‌شوند. اگر گیرنده، قطعات ۴ تا ۷ را بافر کرده باشد به محض دریافت قطعهٔ ۳ می‌تواند دریافت تمام بایتها را تا آخر قطعهٔ هفتم، تأیید کند.

۶-۵-۹ کنترل ازدحام در TCP

هر گاه بار تحویل شده به شبکه بیش از ظرفیتی باشد که می‌تواند از عهدهٔ آن برآید، ازدحام پدید خواهد آمد. اینترنت نیز از این مشکل مستثنی نیست. در این بخش به تشریح الگوریتمهایی خواهیم پرداخت که در طول ربع قرن گذشته برای حل و فصل مشکل ازدحام توسعه یافته‌اند. اگرچه لایهٔ شبکه نیز در مدیریت ازدحام می‌کوشد ولی بار سنگین این مسئولیت بیشتر بر عهدهٔ TCP می‌باشد چرا که راه حل واقعیِ رفع ازدحام، کاهش نرخ ارسال داده‌ها در این لایه است.

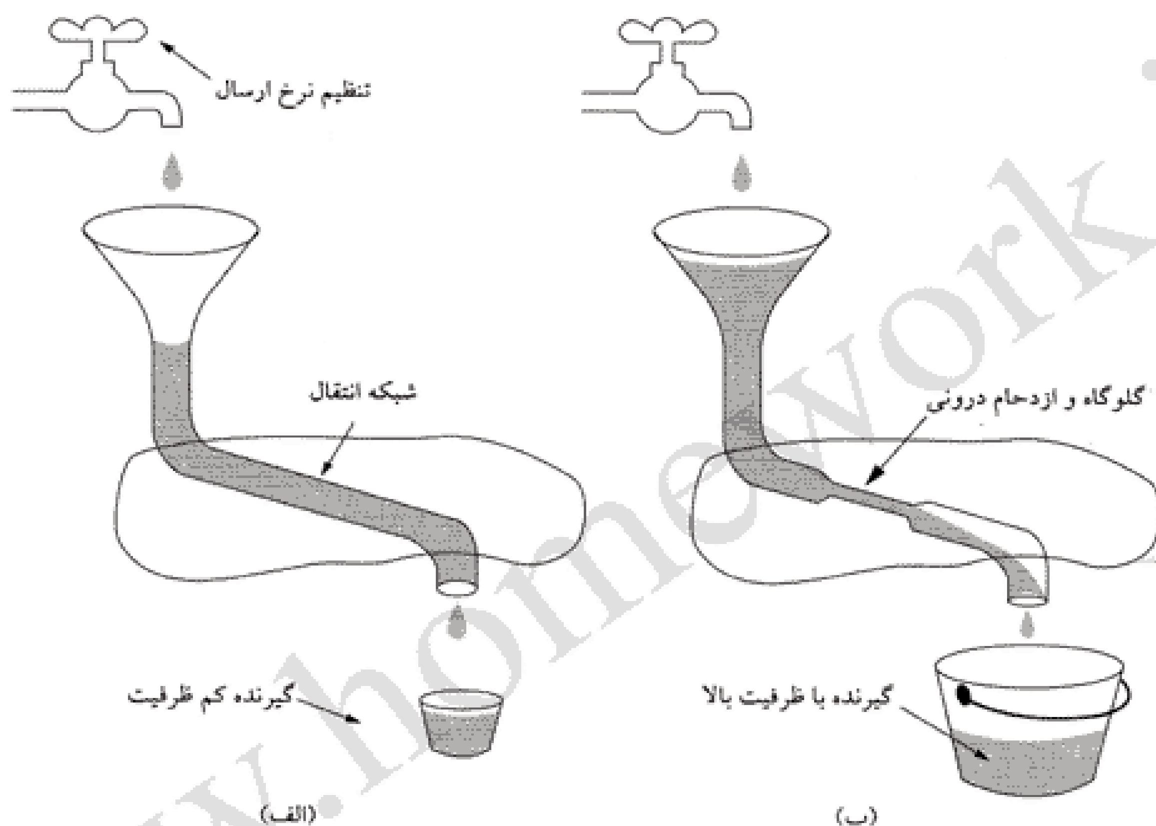
از دیدگاه تئوری، با بکارگیری یک اصل از دانش فیزیک می‌توان به مدیریت ازدحام پرداخت: «**اصل بقای بسته‌ها**»! ایدهٔ اصلی مبنی بر آن است که وقتی بستهٔ قبلی از شبکه خارج شد (یعنی به مقصد تحویل گردید)، بستهٔ جدیدی به شبکه تزریق شود. TCP سعی می‌کند با تنظیم پویا و خودکار اندازهٔ پنجره، بدین هدف نایل آید.

اولین گام در مدیریت ازدحام، تشخیص آن است. در روزگاران گذشته، تشخیص ازدحام بسیار دشوار بود. در آن دوران، به دو دلیل بسته‌ای از دست می‌رفت و مهلت فرستنده منقضی می‌شد: (۱) نویز روی خطوط انتقال (که می‌توانست بسته‌ها را نابود کند) (۲) حذف بسته‌ها توسط مسیریاب دچار ازدحام. تشخیص آنکه کدامیک از عوامل فوق منجر به از دست رفتن یک بسته شده، چندان ساده نبود.

امروزه، از بین رفتن بسته‌ها در اثر خطای خطوط انتقال پدیده‌ای بسیار نادر است زیرا اغلب شاه‌راههای ارتباطی از جنس فیبرنوری هستند. (هر چند شبکه‌های بی‌سیم داستان دیگری دارد.) طبعاً می‌توان نتیجه گرفت که انقضای مهلت ارسال در شبکهٔ اینترنت ناشی از ازدحام است. تمام الگوریتمهای TCP در اینترنت فرض را بر آن گذاشته‌اند که انقضای مهلت (Timeout) و عدم وصول بسته به دلیل بروز ازدحام بوده است و به همین دلیل TCP بر زمانهای Timeout (یعنی زمان انقضای مهلت اعلام وصول هر بسته) به دقت نظارت می‌کند.

قبل از تشریح واکنش TCP در برخورد با ازدحام، ابتدا بررسی می‌کنیم که این پروتکل چه تلاشی در پیشگیری از بروز آن می‌کند. پس از برقراری یک اتصال، بایستی اندازهٔ مناسبی برای پنجره انتخاب شود. گیرنده می‌تواند اندازهٔ پنجره را برحسب بافر در اختیار خود، تعیین نماید. اگر فرستنده به اندازهٔ پنجرهٔ گیرنده پایبند باشد مشکلی از بابت سرریز شدن بافر طرف گیرنده پیش نخواهد آمد ولیکن هنوز هم احتمال بروز مشکلات ناشی از ازدحام درون یک شبکه، وجود دارد.

در شکل ۶-۳۶، این مشکل را با تعبیر هیدرولیکی آن به تصویر کشیده‌ایم. در شکل ۶-۳۶-الف ، یک لولهٔ قطور را مشاهده می‌کنیم که به یک گیرندهٔ کم حجم منتهی شده است. مادامی که فرستنده بیش از ظرفیت سطل، آب تولید و ارسال نکند، سطل هم سرریز نخواهد شد. در شکل ۶-۳۶-ب، عامل محدودکننده، ظرفیت سطل نیست بلکه ظرفیت داخلی حمل زیرشبکه، ایجاد محدودیت کرده است. اگر حجم زیاد آب با سرعت بالا به قیف وارد شود، آن را سریعاً پر کرده و آب به هدر خواهد رفت (در این حالت به دلیل سرریزی قیف).



شکل ۶-۳۶. (الف) یک شبکه سریع که یک گیرنده با ظرفیت کم را تغذیه می‌کند. (ب) یک شبکه کُند که یک گیرنده با ظرفیت بالا را تغذیه می‌کند.

راه حل اینترنت آن است که عوامل بروز این دو مشکل بالقوه را پذیرفته و با هر یک از آنها بطور مجزا و مستقل برخورد کنیم.[1] برای این کار هر فرستنده دو پنجره ایجاد می‌نماید: پنجره اول که براساس اعلام گیرندهٔ طرف مقابل ایجاد می‌شود و پنجرهٔ دوم، «**پنجرهٔ ازدحام**» (Congestion Window). هر یک از این پنجره‌ها تعداد بایتهایی را مشخص می‌کنند که فرستنده می‌تواند ارسال کند. تعداد بایتهایی که فرستنده مجاز به ارسال آنهاست ، مینیمم اندازهٔ این دو پنجره است. بنابراین اندازهٔ مؤثر پنجره، مقدار مینیمم آنچیزی است که فرستنده فکر می‌کند صحیح است و آنچه که گیرنده فکر می‌کند آن باید باشد! مثلاً اگر گیرنده عنوان کند که «هشت کیلوبایت بفرست» ولی فرستنده بداند که ارسال بیش از چهار کیلوبایت شبکه را دچار انسداد می‌کند، چهار کیلوبایت ارسال خواهد کرد. برعکس اگر گیرنده اعلام کند که «هشت کیلوبایت بفرست» و فرستنده نیز بداند که ارسال تا ۳۴کیلوبایت بلامانع است، فقط هشت کیلوبایت درخواستی را ارسال خواهد کرد.

۱. تفکیک عامل «ظرفیت شبکه» از «ظرفیت گیرنده»

وقتی اتصال برقرار می‌شود، فرستنده، اندازهٔ «پنجرهٔ ازدحام» را با طول حداکثر هر قطعه که در حین اتصال توافق شده، مقداردهی اولیه می‌کند؛ سپس یک قطعه با طول حداکثر می‌فرستد. اگر اعلام وصول این قطعه قبل از انقضای مهلت مقرر دریافت شد، اندازهٔ پنجرهٔ ازدحام را به اندازهٔ طول حداکثر قطعه اضافه می‌کند و دفعهٔ بعدی معادل دو قطعه ارسال می‌نماید. مادامی که پس از ارسال هر قطعه دریافت آن تصدیق می‌شود، به اندازهٔ «پنجرهٔ ازدحام» معادل با طول حداکثر هر قطعه اضافه خواهد شد. وقتی اندازهٔ پنجرهٔ ازدحام معادل با طول n قطعه باشد و تمام قطعات ارسالی سر موعد اعلام وصول شوند به پنجرهٔ ازدحام معادل با طول کل n قطعه (برحسب بایت) اضافه خواهد شد. کوتاه سخن آن که اگر در هر بار ارسال (معادل با n قطعه) تمام آنها اعلام وصول شوند، طول پنجرهٔ ازدحام دو برابر می‌شود.

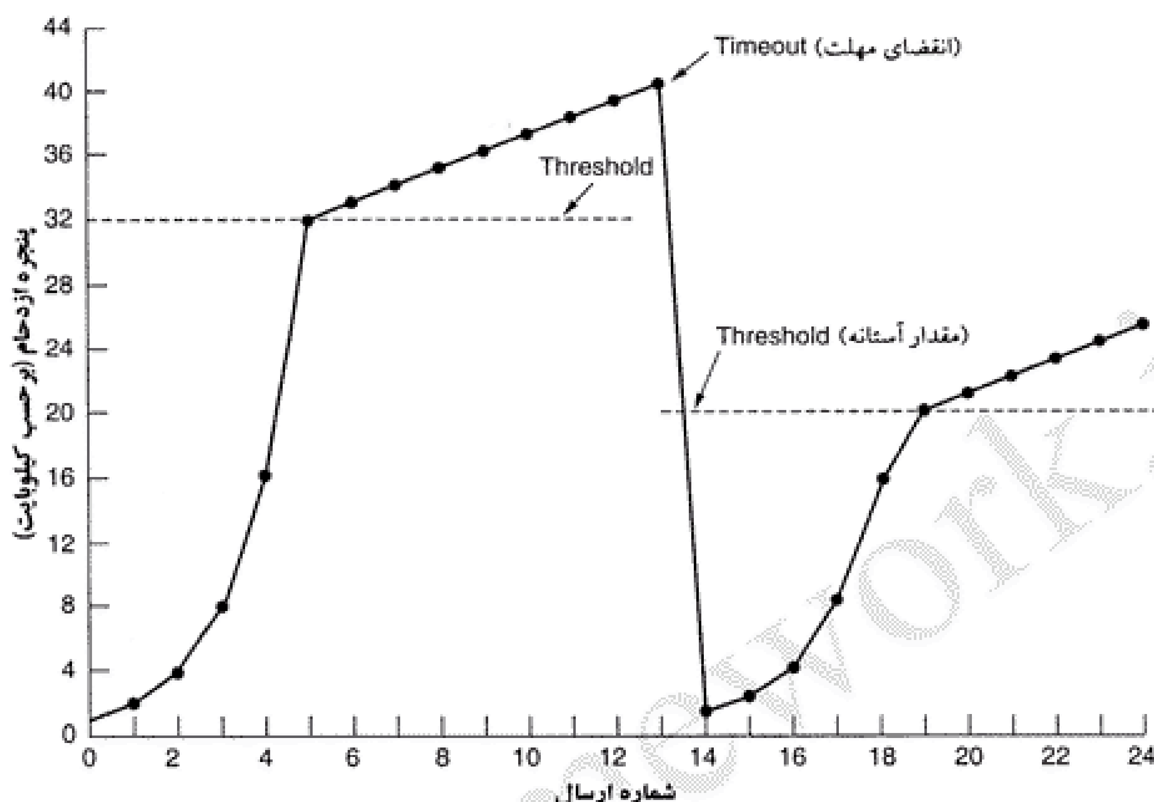
اندازهٔ «پنجرهٔ ازدحام» آنقدر به صورت نمایی رشد می‌کند تا آنکه پس از ارسال قطعات، یا وصول آنها تصدیق نشود و یا آنکه به اندازهٔ پنجرهٔ گیرنده برسد. به عنوان مثال اگر اندازهٔ پنجرهٔ ۱۰۲۴ بایتی، ۲۰۴۸ بایتی، ۴۰۹۶ بایتی به درستی عمل کند ولی ارسال چندین قطعه معادل با ۸۰۹۶ بایت، با مشکل انقضای مهلت اعلام وصول (Timeout) مواجه شود برای اجتناب از ازدحام، اندازهٔ پنجره به ۴۰۹۶ بایت تنظیم می‌شود. مادامی که پنجرهٔ ازدحام بر روی ۴۰۹۶ ثابت می‌ماند حتی اگر طول پنجرهٔ اعلام شده توسط گیرنده از ۴۰۹۶ بیشتر باشد، فرستنده هیچگاه قطعاتی را که مجموع طول آنها از ۴۰۹۶ بیشتر می‌شود نخواهد فرستاد. این الگوریتم اصطلاحاً **«شروع آهسته»** (Slow Start) نام گرفته ولیکن هرگز کُند عمل نمی‌کند زیرا روند آن نمایی است. (Jocobson, 1988) تمام پیاده‌سازیهای TCP، ملزم به حمایت از این الگوریتم هستند.

حال اجازه بدهید به الگوریتم کنترل ازدحام در اینترنت بپردازیم: در اینترنت به غیر از پنجرهٔ گیرنده و پنجرهٔ ازدحام، از پارامتر سومی به نام **«آستانه»** (Threshold) استفاده می‌شود. مقدار اولیهٔ این پارامتر 64KByte است. وقتی پس از ارسال قطعاتی، مهلت اعلام وصول آنها منقضی شود، پارامتر **«آستانه»** به نصف مقدار پنجرهٔ ازدحام تنظیم شده و مقدار پنجرهٔ ازدحام مجدداً به مقدار طول حداکثر یک قطعه بر می‌گردد. حال مجدداً الگوریتم «شروع آهسته» (Slow Start) برای تعیین اندازهٔ مناسب طول پنجرهٔ ازدحام شروع به کار می‌نماید، با این تفاوت که رشد نمایی به محض رسیدن به مقدار آستانه، متوقف می‌شود. پس از رسیدن به نقطهٔ آستانه، ارسال موفق یک قطعه اندازهٔ پنجره را دو برابر نمی‌کند بلکه آن را به صورت خطی افزایش می‌دهد. طبعاً این الگوریتم حدس می‌زند که کاهش طول پنجره به نصف معقول است، فلذا از این نقطه شروع کرده و اندازهٔ آنرا به تدریج افزایش می‌دهد.

برای آن که مشخص شود این الگوریتم کنترل ازدحام چگونه کار می‌کند به شکل ۶-۳۷ نگاه کنید. در اینجا اندازهٔ حداکثر هر قطعه، ۱۰۲۴ بایت است. در ابتدا، اندازهٔ «پنجرهٔ ازدحام»، ۶۴کیلوبایت بوده ولی به دلیل انقضای مهلت اعلام وصول قطعه‌ها (Timeout)، مقدار آستانه به ۳۲کیلوبایت تنظیم شده و اندازهٔ پنجرهٔ ازدحام برای ارسال شمارهٔ صفر به مقدار ۱۰۲۴ کاهش یافته است. از آن به بعد پنجرهٔ ازدحام بطور نمایی رشد کرده تا به مقدار آستانه (یعنی 32KB) رسیده است. پس از آن رشد اندازهٔ پنجرهٔ ازدحام به صورت خطی بوده است.

ارسال سیزدهم موفق نبوده است و مجدداً مشکل عدم اعلام وصول داده‌ها در مهلت مقرر، بروز کرده است. (بروز مشکل در این لحظه چندان دور از ذهن نیست.) لذا مجدداً مقدار آستانه به نصف اندازهٔ پنجرهٔ فعلی تنظیم شده است. (در اینجا اندازهٔ پنجره ۴۰کیلوبایت بوده فلذا نصف آن، ۲۰ کیلوبایت می‌شود.) بار دیگر الگوریتم «شروع آهسته» کار خود را آغاز می‌کند. از آنجایی که در ارسال چهاردهم تا بیستم، پیغام ACK برگشته لذا در هر ارسال اندازهٔ پنجرهٔ ازدحام دو برابر و از آن به بعد رشد پنجره خطی شده است.

اگر هیچگاه مهلت اعلام وصول منقضی نشود، پنجرهٔ ازدحام، رشد خود را آنقدر ادامه می‌دهد تا به اندازهٔ پنجرهٔ گیرنده برسد. در این نقطه، رشد پنجره متوقف شده و مادامی که طول پنجرهٔ گیرنده تغییر نکند و مهلت اعلام



شکل ۶-۳۷. مثالی از الگوریتم کنترل ازدحام در اینترنت.

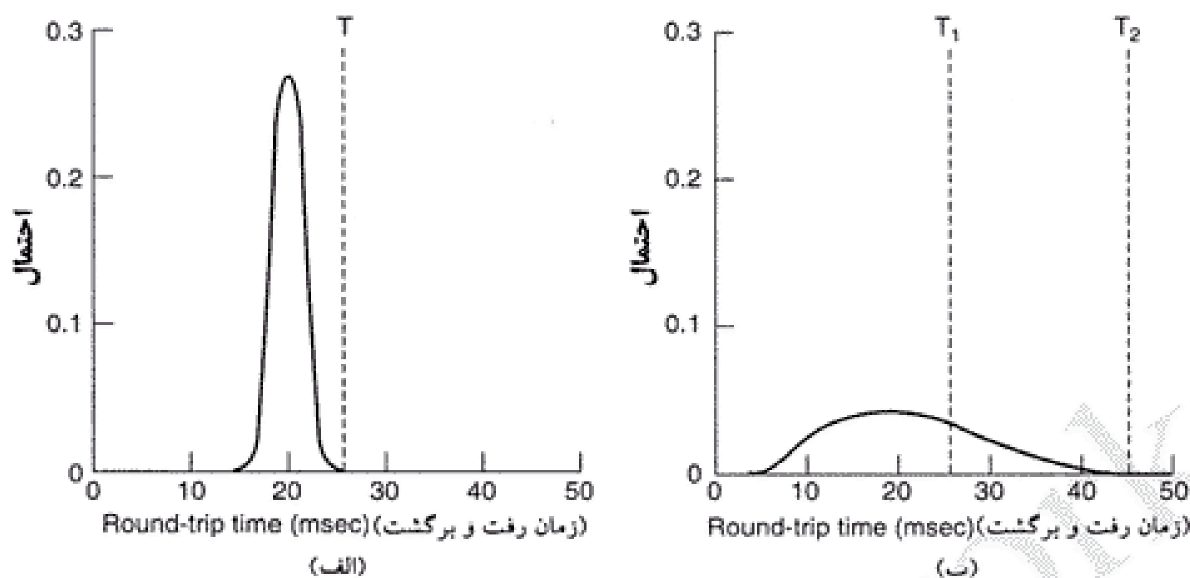
وصول قطعات ارسالی منقضی نشود، اندازهٔ پنجرهٔ ازدحام ثابت خواهد ماند. مضاف بر این اگر پیغام ICMP SOURCE QUENCH دریافت و تحویل TCP شود این رخداد نیز به مثابهٔ انقضای مهلت تلقی می‌شود. راهکار جدیدتر در RFC 3168 تشریح شده است.

۶-۵-۱۰ مدیریت تایمرها در TCP

TCP برای آن که بتواند کارش را بدرستی انجام بدهد از چندین تایمر بهره می‌گیرد. مهمترین آنها، **«تایمر ارسال مجدد»** (Retransmission Timer) است. وقتی قطعه‌ای ارسال می‌شود این تایمر شروع به کار می‌کند. اگر قبل از آن که مهلت این تایمر منقضی گردد، قطعهٔ ارسالی اعلام وصول شود (یعنی Ack آن برگردد)، تایمر متوقف می‌شود ولیکن اگر وصول داده‌ها قبل از صفر شدن مقدار این تایمر تأیید نشود، این قطعه از نو ارسال و تایمر مجدداً راه‌اندازی می‌گردد. حال سؤال مهمی پیش می‌آید: زمان انقضای مهلت چقدر باید باشد؟

این مسئله در لایهٔ انتقال از شبکهٔ اینترنت در مقایسه با همین مسئله در پروتکلهای لایهٔ پیوند داده (فصل سوم)، دشوارتر و پیچیده‌تر است. در لایهٔ پیوند داده، تأخیرها کاملاً قابل تخمین هستند (یا به عبارتی واریانس تأخیر پایین است) فلذا به نحوی که در شکل ۶-۳۸-الف مشاهده می‌شود می‌توان تایمر را به همان مقداری تنظیم کرد که انتظار می‌رود اعلام وصول بسته‌ها (یعنی Ack) قبل از این زمان برگردد. از آنجایی که در لایهٔ پیوند داده‌ها، پیغامهای Ack با تأخیر پیش‌بینی نشده مواجه نمی‌گردد (زیرا مشکل ازدحام وجود ندارد) فلذا عدم دریافت Ack، عموماً به معنای نابودی فریم ارسالی یا نابودی Ack بازگشتی تلقی می‌شود.

TCP با محیطی کاملاً متفاوت مواجه است. «تابع چگالی احتمال زمان بازگشت Ack» به جای آن که شبیه به شکل ۶-۳۸-الف باشد بیشتر شبیه به شکل ۶-۳۸-ب است. تعیین زمان رفت قطعهٔ داده و بازگشت Ack آن



شکل ۶–۳۸. (الف) چگالی احتمال زمان دریافت پیغامهای تصدیق وصول (ACK) در لایهٔ پیوند داده.
(ب) چگالی احتمال زمان دریافت پیغامهای تصدیق وصول در TCP (لایهٔ انتقال).

(Round Trip Time) پیچیده است. حتی اگر این زمان مشخص باشد، تصمیم‌گیری در خصوص مهلت دریافت Ack (یعنی مقدار اولیه تایمر) بسیار دشوار است. اگر زمان انقضای مهلت (یا به عبارتی مقدار اولیه تایمر ارسال مجدد) کوتاه در نظر گرفته شود (مثلاً مقدار T_1 در شکل ۶–۳۸–ب) آنگاه برخی از قطعات داده، بیهوده ارسال مجدد شده و اینترنت با بسته‌های زائد شلوغ می‌شود. برعکس اگر این مقدار بسیار طولانی در نظر گرفته شود (مثلاً مقدار T_2) آنگاه وقتی بسته‌ای از بین می‌رود به دلیل تأخیر بسیار زیادی که در ارسال مجدد بسته پیش می‌آید، کارآیی بشدت افت می‌کند. مضاف بر این مقدار میانگین و واریانس زمان اعلام وصول بسته‌ها می‌تواند در عرض چند ثانیه تغییر جدی داشته باشد. (این تغییر می‌تواند در اثر بروز ازدحام یا رفع آن باشد.)

راه حل نهایی، استفاده از یک الگوریتم کاملاً پویا (Dynamic) است به گونه‌ای که مدام مهلت بازگشت Ack بسته‌ها را تخمین زده و تنظیم نماید. این کار براساس اندازه‌گیری دائمی کارآیی شبکه انجام می‌گیرد. الگوریتمی که عموماً در TCP بکار گرفته می‌شود توسط «ژاکوبسن» (۱۹۸۸) پیشنهاد شده و به نحو زیر کار می‌کند: TCP برای هر اتصال یک متغیر به نام RTT نگاه می‌دارد که مقدار آن بهترین تخمین از زمان رفت بسته به مقصد مورد نظر و بازگشت Ack آن می‌باشد. وقتی یک قطعهٔ داده ارسال می‌گردد یک تایمر شروع به کار می‌کند تا اولاً مشخص شود بازگشت Ack چقدر طول می‌کشد، ثانیاً در صورتی که مهلت مقرر منقضی شد آنرا از نو ارسال نماید. اگر قبل از انقضای مهلت مقرر، دریافت قطعه تصدیق شود، TCP مقدار زمان رفت قطعه و برگشت Ack آنرا، (بکمک مقدار فعلی تایمر RTT) اندازه‌گیری می‌کند. این مقدار را M بنامید. سپس براساس M، مقدار جدید RTT را طبق فرمول زیر بهنگام می‌نماید:

$$RTT = \alpha . RTT + (1 - \alpha) . M$$

در فرمول فوق، α ضریب هموارسازی (Smoothing Factor) است و تعیین می‌کند که مقدار قبلی RTT با چه وزنی در محاسبهٔ مقدار جدید حضور دارد. α عموماً 7/8 است.

حتی با در اختیار داشتن مقدار RTT، انتخاب مقدار مناسب برای تایمر «ارسال مجدد» چندان ساده نیست. عموماً TCP مقدار تایمر را $\beta . RTT$ در نظر می‌گیرد ولیکن مسئله اصلی انتخاب β است. در پیاده‌سازیهای اولیهٔ

TCP، مقدار β همیشه ۲ در نظر گرفته می‌شد ولیکن تجربه نشان داد که انتخاب مقدار ثابت برای β مقدار مناسب و منطقی نیست زیرا با افزایش واریانس، به درستی جواب نمی‌دهد.

در سال ۱۹۹۸ «ژاکوبسن» پیشنهاد کرد که β متناسب با «انحراف معیار تابع چگالی احتمال زمان دریافت پیغامهای Ack»[۱] در نظر گرفته شود. بدین طریق اگر واریانس بالا باشد، β نیز زیاد خواهد بود (و بالعکس). همچنین او پیشنهاد کرد که برای بدست آوردن مقدار تخمینی انحراف معیار از «میانگین انحراف» استفاده شود. به همین دلیل در الگوریتم او به متغیر جدیدی به نام D نیاز است که مقدار تخمینی و هموارشدهٔ (Smoothed) «انحراف» را نگه می‌دارد: وقتی پیغام Ack یک قطعهٔ داده دریافت می‌شود، اختلاف بین مقدار مورد انتظار (یعنی RTT) و مقدار اندازه‌گیری شده (یعنی M) به صورت |RTT-M| محاسبه می‌شود. سپس مقدار هموار شدهٔ D طبق رابطه زیر محاسبه می‌شود:

$$D = \propto.D+(1 - \propto).|RTT-M|$$

مقدار $\propto$ در فرمول بالا می‌تواند با مقدار $\propto$ در فرمول RTT یکسان باشد یا فرق کند. اگرچه مقدار D دقیقاً معادل با انحراف معیار نیست ولیکن به آن نزدیک است و برای محاسبات ما کفایت می‌کند. ژاکوبسن نشان داد که می‌توان فرمول بالا را با جمع، تفریق و شیفت چند عدد صحیح، پیاده‌سازی کرد. اغلب پیاده‌سازیهای عملی TCP از همین الگوریتم بهره گرفته‌اند و زمان انقضای مهلت (یعنی مقدار تایمر ارسال مجدّد) را به صورت زیر تنظیم می‌کنند:

$$Timeout = RTT+4\times D$$

انتخاب ضریب ۴ اختیاری است ولیکن استفاده از این عدد دو مزیت دارد: اول آن که ضرب یک عدد صحیح در ۴ را می‌توان با یک عمل شیفت (دو بیت شیفت به چپ) انجام داد. دوم آن که از انقضای مهلت و ارسال مجدد و غیرضروری جلوگیری می‌کند زیرا کمتر از یک درصد از بسته‌ها با تأخیری بیش از چهار برابر انحراف معیار دریافت می‌شوند. [به عبارت دیگر احتمال آن که بسته‌ای بیهوده ارسال مجدد شود کمتر از یک درصد است.] (در واقع، پیشنهاد ژاکوبسن عدد ۲ بود ولی تجربه نشان داد که عدد ۴، کارآمدتر و بهینه‌تر است.)

مشکلی که در محاسبهٔ پویای RTT بروز می‌کند آن است که وقتی پس از ارسال یک قطعه، مهلت اعلام وصول آن منقضی و قطعه از نو ارسال شد چه باید کرد؟ وقتی که پس از ارسال مجدد یک قطعه، Ack آن دریافت شد روشن نیست که آیا این Ack مربوط به قطعه اول است که دیر رسیده، یا آنکه واقعاً مربوط به قطعهٔ دوم است. حدس اشتباه در این مورد می‌تواند، منجر به غلط شدن نتیجهٔ تخمین RTT شود. شخصی به نام Phil Karn به این مسئله پی برد. وی یک آماتور علاقمند به سیستمهای رادیویی است که می‌خواست بسته‌های TCP/IP را به کمک کانالهای رادیویی که بشدت نامطمئن و توام با خطا هستند ارسال نماید. (در شرایط خوب فقط نیمی از بسته‌ها سالم دریافت می‌شوند!) وی یک پیشنهاد ساده ارائه داد: برای بسته‌هایی که از نو ارسال می‌شوند، مقدار RTT بهنگام نگردد. در عوض مهلت دریافت پیغام Ack، به دو برابر افزایش یابد. این اصلاحیه به نام الگوریتم Karn مشهور است و در پیاده‌سازی TCP از آن استفاده می‌شود.

«تایمر ارسال مجدد» (Retransmission Timer)، تنها تایمری نیست که TCP از آن استفاده می‌کند. تایمر دومی نیز به نام Persistent Timer وجود دارد. از این تایمر برای پیشگیری از بن‌بست ذیل استفاده می‌شود: فرض کنید گیرنده با ارسال Ack اندازهٔ پنجرهٔ خود را به فرستنده صفر اعلام کرده و از او می‌خواهد که منتظر بماند. بعداً گیرنده اندازهٔ جدید پنجرهٔ خود را بهنگام‌سازی و اعلام می‌کند ولیکن بستهٔ حاوی مقدار جدید از بین می‌رود. حال گیرنده و فرستنده هر دو در انتظار یکدیگر به سر می‌برند. برای آن که این انتظار تا ابد ادامه نیابد، به محض

---

۱. Standard Deviation of Acknowledgement Arrival Time Probability Density Function

انقضای مهلت تایمر، فرستنده با ارسال بسته‌ای به گیرنده ، سعی می‌کند او را بیازماید. در پاسخ بدین بسته، اندازهٔ پنجره اعلام خواهد شد. اگر این مقدار کماکان صفر باشد، تایمر مجدداً تنظیم شده و این روند تکرار می‌شود ولی اگر این مقدار غیرصفر باشد فرستنده می‌تواند داده‌های خود را ارسال نماید.

تایمر سومی که در برخی از پیاده‌سازیهای عملی TCP از آن استفاده شده، Keepalive Timer نام دارد. وقتی اتصالی برای مدت زمان طولانی بیکار بماند و مهلت این تایمر منقضی شود، یکی از طرفین سعی می‌کند فعال بودن طرف مقابل خود را بررسی کند. اگر طرف مقابل پاسخی ندهد، اتصال قطع می‌شود. این ویژگی اندکی مناقشه‌برانگیز شده است زیرا اولاً سربار تحمیل می‌کند و ثانیاً ممکن است به دلیل وقفهٔ موقت در شبکه به یک اتصال سالم و صحیح خاتمه داده شود.

آخرین تایمر مورد استفاده در هر اتصال TCP، تایمریست که در وضعیت TIMEED WAIT (شکل ۶-۳۳) در حین بستن یک اتصال، مورد استفاده قرار می‌گیرد. پس از آنکه یک اتصال قطع می‌شود به کمک این تایمر به مدت دو برابر حداکثر طول عمر بسته‌ها، اجازهٔ ایجاد اتصالی با همین شماره پورت را نخواهد داد تا اطمینان حاصل شود که بسته‌های سرگردان (حاصل از اتصال قبل) کاملاً از بین رفته‌اند.

۶-۵-۱۱ TCP و UDP بی‌سیم

از دیدگاه تئوری، پروتکلهای لایهٔ انتقال باید مستقل از تکنولوژی بکار رفته در لایهٔ شبکه باشند. به ویژه، نباید برای TCP اهمیت داشته باشد که IP بر روی فیبرنوری کار می‌کند یا بر روی کانالهای رادیویی، ولیکن در عمل این موضوع اهمیت دارد زیرا پیاده‌سازی عملی TCP اغلب بر مبنای مفروضاتی بهینه‌سازی می‌شود که فقط برای شبکه‌های سیمی صادق هستند و برای شبکه‌های بی‌سیم با شکست مواجه می‌گردد. نادیده انگاشتن ویژگیهای انتقال بی‌سیم می‌تواند به نوعی از پیاده‌سازی TCP منتهی شود که از دیدگاه منطقی درست است ولی عملاً کارآیی بسیار پایینی دارد.

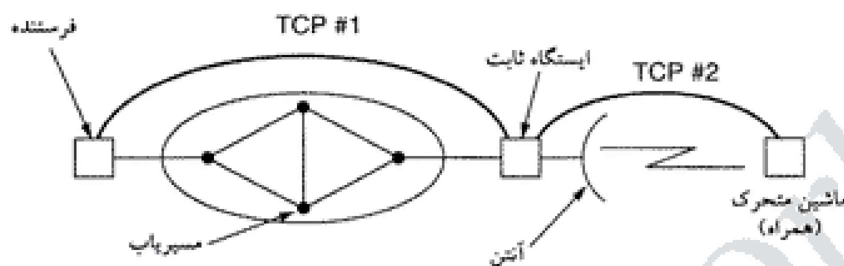
بنیادی‌ترین مشکل در الگوریتم کنترل ازدحام بروز می‌کند. امروزه تقریباً در تمام پیاده‌سازیهای عملی TCP فرض شده که انقضای مهلت (Timeouts) در اثر ازدحام بوده است و طبعاً خطای کانال انتقال را نادیده می‌گیرند. در نتیجه وقتی تایمر به صفر می‌رسد و مهلت دریافت Ack منقضی می‌شود، TCP سرعت خود را پایین آورده و با شدت کمتری ارسال می‌کند. ایده‌ای که در پشت این راهکار نهفته است کاهش بار شبکه و کم کردن ازدحام می‌باشد.

متأسفانه لینکهای ارتباطی بی‌سیم بسیار غیرقابل اعتماد و توأم با خطا هستند و همیشه برخی از بسته‌ها را از بین می‌برند. در اینجا راهکار مناسب در مواجهه با بسته‌های نابود شده آنست که در اسرع وقت از نو ارسال شوند. کاهش سرعت ارسال، وضع را به مراتب بدتر می‌کند. اگر مثلاً بیست درصد از کل بسته‌ها از بین بروند ، وقتی فرستنده ۱۰۰ بسته در ثانیه ارسال می‌کند درصد مفید آن ۸۰ بسته در ثانیه خواهد بود. در چنین حالتی کاهش نرخ ارسال بسته به ۵۰ بسته در ثانیه، ظرفیت مفید خروجی را به ۴۰ بسته در ثانیه کاهش خواهد داد.

در نتیجه وقتی بسته‌ای در یک شبکهٔ سیمی از دست می‌رود باید سرعت ارسال فرستنده کاهش یابد ولیکن وقتی همین اتفاق در یک شبکهٔ بی‌سیم می‌افتد نه تنها فرستنده نباید سرعت خود را پایین بیاورد بلکه باید تلاش بیشتری در ارسال سریع و مجدد آنها داشته باشد. وقتی که فرستنده از ماهیت شبکهٔ خود آگاه نیست تصمیم‌گیری درست دشوار است.

بسیار اتفاق می‌افتد که مسیر بین فرستنده و گیرنده ناهمگون است یعنی ممکن است ۱۰۰۰ کیلومتر از مسیر در شبکه‌ای سیمی و ۱ کیلومتر آخر بی‌سیم باشد. در چنین حالتی تصمیم‌گیری در خصوص مهلت دریافت Ack به مراتب دشوارتر است زیرا محل بروز مشکل اهمیت دارد. [اگر بسته‌ای در شبکهٔ سیمی از بین رفته باشد ناشی از

مشکل ازدحام و اگر در شبکهٔ بی‌سیم خراب شده باشد ناشی از خطای کانال بوده است.] برای حل این مشکل راه حلی توسط Bakne و Badrinath (۱۹۹۵) ارائه شده که «TCP غیرمستقیم» (indirect TCP) نام دارد. در این روش، یک اتصال TCP به نحوی که در شکل ۶-۳۹ دیده می‌شود به دو اتصال مجزا تقسیم می‌شود: اتصال اول بین فرستنده و ایستگاه ثابت (Base Station) برقرار می‌شود. اتصال دوم بین ایستگاه ثابت و گیرنده شکل می‌گیرد. ایستگاه ثابت به سادگی بسته‌ها را در هر دو جهت، بین این دو اتصال کپی می‌کند. [به عبارتی از اتصال اول دریافت و بر روی اتصال دوم ارسال می‌کند و بالعکس]



شکل ۶-۳۹. تقسیم یک اتصال TCP به دو اتصال مجزا.

مزیت این روش آن است که هر دو اتصال در شبکه‌ای همگن شکل می‌گیرند. انقضای مهلت در اتصال اول می‌تواند از سرعت ارسال بکاهد ولیکن بروز همین مشکل در اتصال دوم می‌تواند به افزایش سرعت فرستنده بینجامد. پارامترهای دیگر نیز براساس محیط هر یک از این دو اتصال به صورت مستقل و پویا تنظیم می‌شوند. اشکال این روش نیز آن است که مفهوم TCP را نقض می‌کند. از آنجایی که هر یک از بخشهای یک اتصال خودش یک اتصال کامل است فلذا وقتی دریافت داده‌ها به فرستنده اعلام می‌شود به معنای آن نیست که گیرندهٔ حقیقی آنها را دریافت کرده است بلکه به معنای دریافت آنها توسط ایستگاه ثابت می‌باشد.

راه حل دیگری نیز توسط Balakrishnan و همکاران او پیشنهاد شده که مفهوم TCP را نقض نمی‌کند. (۱۹۹۵) روش آنها مستلزم ایجاد تغییرات کوچکی در کُد اجرایی لایهٔ شبکه در ایستگاه ثابت است. یکی از تغییرات، افزودن یک «**عامل تحقیق و تفحص**» (Snooping Agent) به لایهٔ شبکه از ایستگاه ثابت می‌باشد؛ این عامل تمام قطعات TCP را که به سوی یک ماشین متحرک بی‌سیم می‌روند یا بسته‌های حاوی Ack بازگشتی از آنرا تشخیص داده و موقتاً در حافظهٔ نهان (cache) خود ذخیره می‌نماید. اگر «عامل تحقیق و تفحص» متوجه شود که قطعه‌ای برای ماشین متحرک ارسال شده ولی Ack آن در مدت زمان معقول و کوتاهی برنگشته، آن قطعه را بدون اطلاع دادن به فرستندهٔ آن از نو ارسال می‌کند. همچنین وقتی یک Ack تکراری از ماشین متحرک می‌بیند متوجه می‌شود که این ماشین چیزی را از دست داده است. Ackهای تکراری نیز در «عامل تحقیق و تفحص» حذف می‌شوند تا ماشین مبداء، دریافت آنها را اشتباهاً بمعنای بروز ازدحام تعبیر نکند. در این روش سعی می‌شود ناهمگونی شبکه‌ها به نحو زیرکانه‌ای از چشم طرفین پنهان بماند.

یک اشکال در این روش نامرئی آنست که اگر لینک بی‌سیم بسیار پر خطا باشد و درصد بالایی از بسته‌ها را نابود کند، ممکن است ماشین مبداء به دلیل عدم دریافت Ack در مهلت مقرر، الگوریتم کنترل ازدحام خود را فراخوانی و اعمال نماید. در روش «TCP غیرمستقیم» الگوریتم کنترل ازدحام هرگز شروع نخواهد شد مگر آنکه واقعاً در بخش سیمی شبکه ازدحام رخ داده باشد.

در مقالهٔ Balakrishnan و همکاران او، راه حلی نیز برای مشکل از بین رفتن بسته‌های ماشین متحرک (بی‌سیم) ارائه شده است: وقتی ایستگاه ثابت متوجه می‌شود که در بین داده‌های ارسالی از ماشین متحرک یک

قطعهٔ جا افتاده وجود دارد، با استفاده از یک گزینهٔ جدید در فیلد Option از قطعهٔ TCP، تقاضایی را تولید کرده و جهت دریافت این قطعهٔ جا افتاده برای ماشین متحرک می‌فرستد. به کمک این اصلاحیه، لینک بی‌سیم در دو جهت قابل اعتمادتر می‌شود بدون آن که مبداء چیزی در این خصوص بداند و بی‌آن که مفهوم واقعی TCP نقض شود.

اگرچه UDP از مشکلاتی که TCP با آن مواجه است، رنج نمی‌برد ولی UDP نیز در محیطهای بی‌سیم مشکلات خاص خود را دارد. مشکل عمده آن است که برنامه‌های کاربردی استفاده‌کننده از UDP انتظار قابلیت اعتماد بالا دارند. اگرچه UDP تحویل داده‌ها را تضمین نکرده ولیکن انتظار می‌رود که حتی‌الامکان درست عمل کند. [یعنی درصد بسیار ناچیزی از داده‌ها از بین برود.] در محیط بی‌سیم، UDP بسیار نامطمئن و پرخطا عمل خواهد کرد. اگر آن دسته از برنامه‌های کاربردی که می‌توانند پیامهای UDP گمشده را تشخیص داده و بازیابی کنند، از محیطی که در آن احتمال از بین رفتن بسته‌ها ناچیز است به محیطی که بخشی از داده‌ها بطور دائم از دست می‌روند، منتقل شوند منجر به کاهش بحرانی کارآیی آنها خواهد شد.

ارتباط بی‌سیم، جدای از مسئله کارآیی، در زمینه‌های دیگری نیز تأثیر منفی دارد. به عنوان مثال چگونگی پیدا کردن یک چاپگر محلی و برقراری اتصال با آن، خود یک مسئله است. یا مشکل دیگر آنست که چگونه می‌توان به صفحه وب محلی در سلول جاری دسترسی پیدا کرد در حالی که نام آن را نمی‌دانیم. معمولاً طراحان صفحهٔ وب [در محیطهای شبکه محلی] فرض را بر آن می‌گذارند که پهنای باند موجود فراوان است. حال اگر در هر صفحهٔ وب لوگویی قرار داده شود که مراجعه به آن در یک شبکهٔ بی‌سیم مثلاً ده ثانیه طول می‌کشد، خوشایند کاربر نخواهد بود.

شبکه‌های بی‌سیم در حال رواج هستند و مشکل اجرای TCP بر روی اینگونه شبکه‌ها روز به روز حادتر می‌شود. در این خصوص کارهای دیگری نیز انجام شده که گزارش آنها در مراجع زیر در دسترس است:

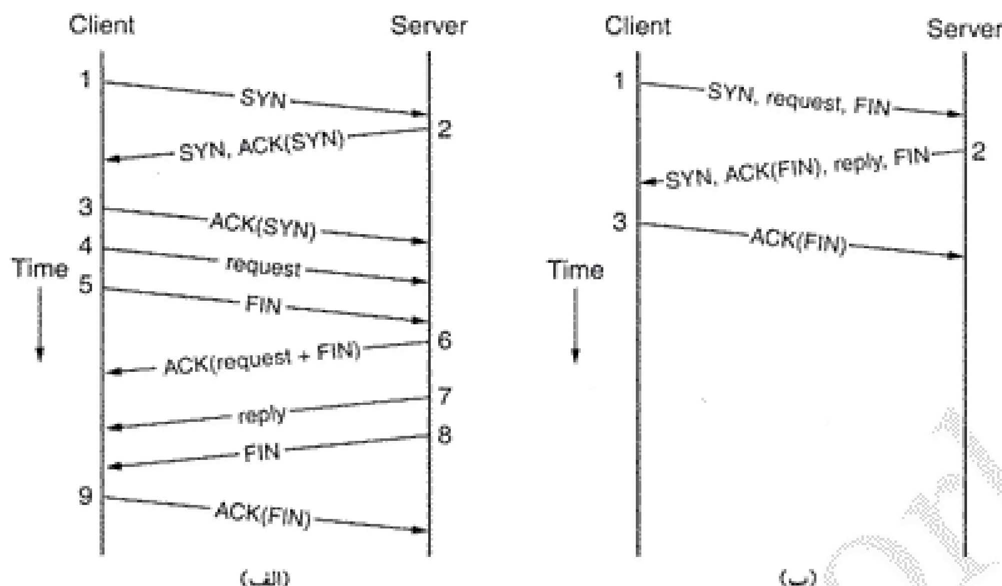
Barakat et al., 2000; Ghani and Dixit, 1999; Huston, 2001; and Xylomenos et al., 2001)

۱۲-۵-۶　TCP تراکنشی (Transactional TCP)

قبلاً در این فصل به موضوع فراخوانی پروسیجرهای راه دور (RPC) به عنوان روشی برای پیاده‌سازی سیستمهای سرویس‌دهنده/مشتری، نگاهی انداختیم. اگر «تقاضا» (request) و «پاسخ» (Reply) آنقدر کوچک باشند که در یک بستهٔ واحد جا بگیرند و تقاضاهای متوالی، مستقل از هم باشند، بسادگی می‌توان از UDP بهره گرفت. ولیکن اگر این شرایط برقرار نباشد، استفاده از UDP چندان مفید نخواهد بود. به عنوان مثال اگر پاسخ به یک تقاضا، داده‌ای بسیار بزرگ باشد، قطعات داده باید شماره‌گذاری شده و مکانیزمی جهت ارسال مجدد قطعات از بین رفته، ابداع و پیاده‌سازی شود. در نتیجه، برنامهٔ کاربردی ملزم به پیاده‌سازی عملیات TCP خواهد بود.

روشن است که این رویکرد جالب نیست ولی استفاده از TCP به جای UDP نیز فایده‌ای ندارد. مشکل استفاده از TCP، کارآیی آن است. اگر از RPC بر روی TCP استفاده شده باشد، توالی بسته‌هایی که بین سرویس‌دهنده و مشتری (برای انجام یک فراخوانی راه دور) مبادله می‌شوند در شکل ۶-۴۰-الف نشان داده شده است. در بهترین حالت باید ۹ بسته مبادله شود. این نه بسته عبارتند از:

۱. برنامهٔ مشتری، یک بستهٔ SYN به منظور برقراری اتصال برای سرویس‌دهنده ارسال می‌کند.
۲. سرویس‌دهنده در پاسخ به دریافت بستهٔ SYN، یک بستهٔ ACK بازپس می فرستد.
۳. برنامهٔ مشتری فرآیند دست تکانی سه مرحله‌ای را تکمیل می‌نماید.
۴. برنامهٔ مشتری، تقاضای اصلی خود را ارسال می‌دارد.
۵. برنامهٔ مشتری، بستهٔ FIN را جهت اعلام خاتمهٔ کار خود می‌فرستد.



شکل ۶-۴۰. (الف) عمل RPC بکمک TCP معمولی. (ب) عمل RPC بکمک T/TCP .

۶. سرویس‌دهنده دریافت بستهٔ حاوی تقاضا و FIN را تصدیق می‌کند. (با ارسال ACK)
۷. سرویس‌دهنده پاسخ مورد نظر مشتری را باز می‌گرداند.
۸. سرویس‌دهنده یک بستهٔ FIN می‌فرستد تا خاتمهٔ کار خود را اعلام نماید.
۹. مشتری، دریافت بستهٔ FIN متعلق به سرویس‌دهنده را تصدیق می‌کند.

دقت کنید که این ۹ مرحله در بهترین حالت ممکن اتفاق می‌افتد. در بدترین حالت، تقاضای مشتری و بستهٔ FIN بطور جداگانه اعلام وصول می‌شود و به همین نحو پاسخ سرویس‌دهنده و بستهٔ FIN آن بطور مجزا ارسال می‌گردد.

سؤالی که مطرح می‌شود آنست که آیا راهی وجود دارد که کارآیی و سرعت RPC مبتنی بر UDP (دقیقاً با دو پیام) و قابلیت اعتماد TCP را با هم تلفیق کرد. پاسخ به این سؤال این است: تقریباً! این کار را می‌توان با یک گونهٔ آزمایشی از TCP که اصطلاحاً T/TCP (Transactional TCP) نامیده می‌شود انجام داد. شرح T/TCP در RFC 1379 و RFC 1644 آمده است.

ایدهٔ اصلی در T/TCP آن است که روند استاندارد برقراری یک اتصال را به گونه‌ای اصلاح کنیم که بتوان در حین برقراری اتصال، داده‌ها را نیز ارسال کرد. پروتکل T/TCP در شکل ۶-۴۰-ب نشان داده شده است. اولین بستهٔ ارسالی توسط مشتری، حاوی بیت SYN، اصل تقاضا و بیت FIN است. در حقیقت می‌گوید: «مایل به برقراری یک اتصال هستم، این هم داده‌های من، کارم نیز به اتمام رسید!»

وقتی سرویس‌دهنده، این تقاضا را دریافت می‌کند، پاسخ آن را پیدا یا محاسبه کرده و سپس چگونگی ارسال آن را انتخاب می‌نماید: اگر پاسخ در یک بستهٔ واحد جا بگیرد، بسته‌ای را که در شکل ۶-۴۰-ب نشان داده شده، ارسال می‌دارد. در حقیقت اعلام می‌کند: «دریافت FIN شما را تأیید می‌کنم، این هم پاسخ شما، کار من هم تمام شد!» در اینجا، مشتری دریافت FIN سرویس‌دهنده را تصدیق کرده و پروتکل با مبادلهٔ سه پیام خاتمه می‌یابد.

ولیکن اگر پاسخ بزرگتر از یک بسته باشد، سرویس‌دهنده می‌تواند بیت FIN را فعال نکند و چندین بسته بفرستد. سپس در آخرین بسته FIN را فعال و ارسال نماید.

اشاره به این نکته ارزشمند است که T/TCP تنها نسخهٔ بهبود یافتهٔ TCP برای عملیات تراکنشی نیست.

پیشنهاد دیگر SCTP (Stream Control Transmission Protocol) است. از ویژگیهای آن می‌توان به موارد ذیل اشاره کرد: (۱) حفظ مرز پیام (۲) پشتیبانی از چندین حالت تحویل (مثل تحویل نامرتب) (۳)حمایت از Multihoming (ماشینهای مقصد پشتیبان) (۴) اعلام وصول بسته‌ها به صورت انتخابی (Selective Repeat). (Stewart and Metz, 2001) با این حال وقتی یک نفر پیشنهاد ایجاد تغییر در پروتکلی را که برای سالها به خوبی کار کرده، می‌دهد یک مناقشهٔ عظیم بین طرفداران این دو نظریه در می‌گیرد: «افرادی که معتقدند باید برای پاسخ به نیاز کاربران ویژگیهای بیشتری را به پروتکل افزود» و «افرادی که معتقدند تا وقتی پروتکل به بن‌بست نخورده نباید آن را اصلاح کرد!»

۶-۶ مسائل مرتبط با کارآیی[۱]

مسائل مرتبط با کارآیی در شبکه‌های کامپیوتری از اهمیت ویژه‌ای برخوردارند. وقتی صدها یا هزاران کامپیوتر به هم متصل می‌شوند، تعامل پیچیدهٔ آنها با تبعات غیرمنتظره‌ای همراه است. اکثراً این پیچیدگی منجر به کارآیی بسیار ضعیف شبکه می‌شود و هیچکس هم نمی‌داند علت چیست. در بخشهای آتی به بررسی مواردی خواهیم پرداخت که به کارآیی شبکه مربوط می‌شود تا ببینیم که چه مشکلاتی وجود دارد و با آنها چه باید کرد.

متأسفانه، تشخیص کارآیی شبکه بیشتر یک هنر است تا یک علم! تئوری کمی وجود دارد که بتوان در عمل از آن بهره گرفت. بهترین کاری که می‌توانیم انجام بدهیم آنست که برخی از قواعد تجربی با پشتوانه محکم و مثالهایی از دنیای واقعی را معرفی کنیم. تعمداً این مبحث را تا اینجا به تعویق انداختیم تا پس از مطالعهٔ TCP بتوانیم از آن به عنوان مثال استفاده نماییم.

لایهٔ انتقال تنها نقطهٔ بروز مشکلات مرتبط با کارآیی نیست. در فصل قبلی برخی از این مشکلات را در لایهٔ شبکه بررسی کردیم. علیرغم این، لایهٔ شبکه بیشتر درگیر مسائل مسیریابی و کنترل ازدحام است. موارد مرتبط با سیستمهای نهایی به لایهٔ انتقال بر می‌گردد لذا این فصل جایگاه مناسبی برای بررسی مشکلات مرتبط با کارآیی است. در پنج بخش بعدی به پنج جنبه از کارآیی شبکه نگاهی خواهیم انداخت:

۱. مشکلات کارآیی
۲. اندازه‌گیری کارآیی شبکه
۳. طراحی سیستم برای رسیدن به کارآیی بهتر
۴. پردازش سریع TPDU
۵. پروتکلهایی برای شبکه‌های کارآمد در آینده

برای هر واحد اطلاعات که توسط لایهٔ انتقال مبادله می‌شود، نیاز به یک اسم عمومی داریم. واژهٔ بکار رفته در TCP یعنی «قطعه» (Segment) گمراه‌کننده است و در هیچ جای دیگر به غیر از پروتکل TCP از این واژه استفاده نشده است. واژه‌های بکار رفته در ATM (مثل CS-PDU ، SAR-PDU و CPCS-PDU) نیز خاص شبکهٔ ATM هستند. واژهٔ «بسته» در لایهٔ شبکه و واژهٔ «پیام» در لایهٔ کاربرد بکار می‌روند. به دلیل فقدان یک استاندارد و اجماع عمومی، ما از همان واژهٔ TPDU استفاده خواهیم کرد. وقتی بخواهیم به بسته و TPDU درون آن بطور همزمان اشاره کنیم واژهٔ کلی «بسته» را بکار خواهیم برد مثلاً وقتی می‌گوییم: «CPU باید آنقدر سریع باشد که بتواند بسته‌های ورودی را به صورت بی‌درنگ پردازش کند» منظورمان هم بستهٔ لایهٔ شبکه و هم TPDU جاسازی شده در آن می‌باشد.

---

۱. Performance

۶-۶-۱ مشکلات کارآیی در شبکه‌های کامپیوتری

منشاء برخی از مشکلات کارآیی مثل ازدحام به استفادهٔ بیش از حد از منابع موجود شبکه برمی‌گردد. اگر به ناگاه ترافیکی بیش از توان و ظرفیت مسیریاب، به آن تحویل داده شود ازدحام پدید آمده و موجب کاهش کارآیی می‌شود. در فصل قبلی مفصلاً به موضوع ازدحام پرداختیم.

همچنین وقتی که منابع شبکه از لحاظ ساختاری توازن و تعادل نداشته باشند کارآیی کاهش خواهد یافت. به عنوان مثال اگر یک خط ارتباطی یک گیگابیتی به یک PC معمولی متصل شده باشد پردازندهٔ ضعیف این کامپیوتر قادر نخواهد بود که بسته‌های ورودی را به سرعت پردازش کند و برخی از آنها از دست می‌رود. بسته‌های از دست رفته نهایتاً از نو ارسال می‌شوند و طبعاً تأخیر افزایش می‌یابد، پهنای باند هدر می‌رود و در مجموع کارآیی شبکه افت می‌کند.

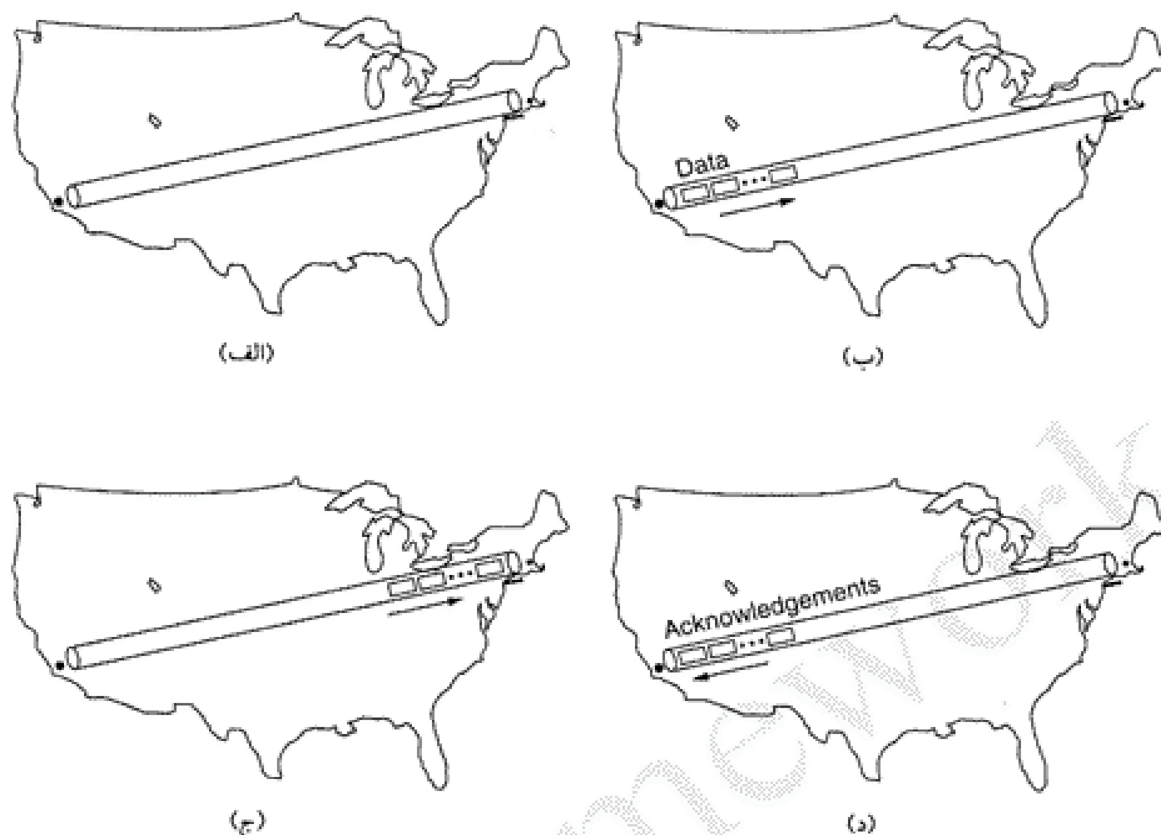
بار بیش از حد (Overload) نیز ممکن است بطور لحظه‌ای اتفاق افتاده و کارآیی شبکه را کاهش بدهد. به عنوان مثال اگر یک TPDU حاوی یک پارامتر اشتباه باشد (مثلاً شمارهٔ پورت مقصد آن صحیح نباشد)، در بسیاری حالات، گیرنده یک گزارش خطا باز می‌گرداند. حال در نظر بگیرید که اگر یک TPDU حاوی پارامتر اشتباه به صورت پخش فراگیر (Broadcast) برای ۱۰۰۰۰ ماشین ارسال شود چه اتفاقی می‌افتد: ممکن است همهٔ آنها یک پیغام خطا برگردانند! این اتفاق منجر به بروز **«طوفان پخش فراگیر»** (Broadcast Storm) شده و می‌تواند شبکه را فلج کند. UDP از این مشکل رنج می‌برد مگر آن که این پروتکل به نحوی تغییر کند که در مواجهه با TPDUهای اشتباه که به صورت فراگیر پخش شده‌اند پیغام خطا برنگرداند.

مثال دوم از تحمیل بار بیش از اندازه به شبکه (بطور همزمان)، پس از قطع برق اتفاق می‌افتد. وقتی برق مجدداً بر می‌گردد همهٔ ماشینها بطور همزمان از طریق ROM خود سعی می‌کنند از نو راه‌اندازی شوند. عموماً در روال راه‌اندازی، هر ماشین شبکه در ابتدا به برخی از سرویس‌دهنده‌ها (مثل DHCP) مراجعه می‌کند تا اول هویت خود را بشناسد؛ پس از آن برای دریافت سیستم عامل به سرویس‌دهندهٔ فایل مراجعه می‌نماید. اگر صدها ماشین به طور همزمان این کار را انجام بدهند، احتمال دارد سرویس‌دهنده در زیر چنین باری از کار بیفتد.

حتی اگر بار، بیش از حد و همزمان اتفاق نیفتد و منابع شبکه نیز بقدر کفایت در اختیار باشد باز هم ممکن است به دلیل عدم تنظیم صحیح سیستم، کارآیی شبکه کاهش یابد. به عنوان مثال اگر ماشینی دارای پردازندهٔ بسیار سریع و حجم انبوه حافظه باشد ولیکن به قدر کافی برای فضای بافر حافظه اختصاص داده نشود ممکن است کمبود بافر به از دست رفتن TPDU بینجامد. همچنین اگر الگوریتم زمان‌بندی پروسه‌ها، اولویت بالایی را به پردازش TPDUها ندهد باز هم ممکن است برخی از بسته‌ها از بین بروند.

مشکل دیگر، تنظیم صحیح مهلت تایمرهاست. وقتی یک TPDU ارسال می‌گردد برای آن که بتوان از گم شدن آن آگاه شد یک تایمر برای آن تنظیم می‌شود. اگر مهلت این تایمر کوتاه در نظر گرفته شود، ارسالهای مجدد بیهوده اتفاق می‌افتد و پهنای باند کانال را هدر می‌دهد. اگر هم این زمان طولانی در نظر گرفته شود در هنگام از دست رفتن یک TPDU تأخیر بی‌مورد تحمیل می‌شود. پارامتر دیگری که باید تنظیم شود مهلت تایمری است که وقتی داده‌ای دریافت می‌شود و نمی‌خواهیم برای آن ACK مجزا بفرستیم باید به آن مدت صبر کرد تا داده‌ای جهت ارسال تولید شود و بتوان ACK را به آنها ضمیمه نمود. اگر این زمان انتظار زیاد در نظر گرفته شود، ارسالهای تکراری و بیهوده اتفاق می‌افتد.

شبکه‌های گیگابیتی مشکلات جدیدی در کارآیی شبکه به همراه آورده‌اند. به عنوان مثال در نظر بگیرید که یک تودهٔ ۶۴ کیلوبایتی داده از سن دیه‌گو به بوستون ارسال می‌شود تا در بافر ۶۴ کیلوبایتی گیرنده قرار بگیرد. فرض کنید تأخیر انتشار این لینک (با احتساب سرعت نور در فیبر) معادل ۲۰ میلی‌ثانیه باشد. در لحظهٔ $t=0$ خط خالی



شکل ۶-۴۱. وضعیت ارسال یک مگابیت داده از سن‌دیه‌گو به بوستون. (الف) در لحظهٔ t=0. (ب) پس از گذشت ۵۰۰ مایکروثانیه. (ج) پس از گذشت ۲۰ میلی ثانیه. (ب) پس از گذشت ۴۰ میلی ثانیه.

است و به نحوی که در شکل ۶-۴۱-الف می‌بینید داده‌ای بر روی خط ارسال نشده است. حدود ۵۰۰ میکروثانیه بعد، تمام TPDUهای حاوی ۶۴ کیلوبایت داده بر روی فیبر قرار گرفته‌اند و ارسال آنها خاتمه یافته است. حالا اولین TPDU ارسالی جایی در نزدیکی براولی (Brawley) است! (شکل ۶-۴۱-ب) با این حال فرستنده باید ارسال خود را تا اعلام مجدد اندازهٔ پنجره متوقف کند چرا که متناسب با اندازهٔ پنجرهٔ گیرنده، داده ارسال شده است.

پس از ۲۰ میلی‌ثانیه، اولین TPDU به بوستون می‌رسد و طبق شکل ۶-۴۱-ج اعلام وصول آنها آغاز می‌شود. ۴۰ میلی‌ثانیه پس از شروع، اولین پیغام ACK به فرستنده باز می‌گردد و دومین تودهٔ داده را می‌توان ارسال کرد. از آنجایی که در طی ۴۰ میلی‌ثانیه، فقط ۰/۵ میلی‌ثانیه از خط انتقال استفاده شده، کارآیی آن حدود ۱/۲۵ درصد است. چنین وضعیتی برای پروتکلهای قدیمی که بر روی خطوط گیگابیتی اجرا می‌شوند، کاملاً طبیعی است.

یکی از کمیتهای بسیار مهمی که در هنگام تحلیل کارآیی شبکه باید به خاطر داشته باشید «**حاصلضرب پهنای باند در تأخیر**» (Bandwidth-Delay Product) است. این کمیت را می‌توان با ضرب مقدار پهنای باند (برحسب بیت بر ثانیه) در زمان تأخیر رفت و برگشت خط (برحسب ثانیه) محاسبه کرد. این حاصلضرب، ظرفیت خط لولهٔ بین فرستنده و گیرنده و بالعکس را برحسب بیت تعیین می‌کند.

به عنوان مثال در شکل ۶-۴۱ حاصلضرب پهنای باند در تأخیر، معادل ۴۰ میلیون بیت است. یعنی فرستنده مجبور است یک تودهٔ ۴۰ میلیون بیتی از داده‌ها را بفرستد تا اولین پیغام ACK باز گردد. این تعداد بیت کل خط را

در دو جهت پر می‌کند.[1] این همان دلیلی است که راندمان خط برای ارسال نیم میلیون بیت (۶۴کیلوبایت) داده به مقدار ۱/۲۵ درصد کاهش می‌یابد زیرا فقط ۱/۲۵ درصد از حجم ۴۰ میلیون بیتی ارسال و متوقف شده است.

نتیجه‌ای که می‌توان گرفت آنست که برای کارآیی خوب ، پنجرهٔ گیرنده حداقل باید به بزرگی «**حاصلضرب پهنای باند در تأخیر**» باشد. (حتی ترجیحاً بیشتر زیرا ممکن است گیرنده نتواند فوراً پاسخ بدهد.) برای خطوط گیگابیتی بین قاره‌ای به حداقل ۵ مگابایت فضای حافظه نیاز است.

اگر کارآیی خط برای ارسال یک مگابیت داده اینقدر پایین باشد تصور کنید که برای یک تقاضای کوتاه چند صد بایتی چقدر است! اگر وقتی اولین مشتری منتظر دریافت پاسخ از طرف مقابل است از خط استفادهٔ دیگری نشود، خط یک گیگابیتی هیچ مزیتی بر خط یک مگابیت بر ثانیه نخواهد داشت؛ فقط گرانتر است.

مشکل دیگر کارآیی، که برای کاربردهای حساس به زمان مثل صدا و تصویر رخ می‌دهد، «لرزش» (Jitter) است. کوتاه بودن زمان متوسط انتقال کافی نیست؛ انحراف معیار این زمان نیز باید پایین باشد. رسیدن به زمان متوسط انتقال کم و انحراف معیار پایین مستلزم تلاشهای مهندسی بسیار جدی است.

۶-۶-۲ اندازه‌گیری کارآیی شبکه

وقتی شبکه ضعیف و ناکارآمد عمل می‌کند کاربران آن شروع به شکوه و گلایه کرده و خواهان بهبود کارآیی می‌شوند. برای بهبود کارآیی، اپراتورها باید اول تعیین کنند که مشکل از کجا منشاء می‌گیرد. بمنظور پیگیری و تشخیص مشکل بوجود آمده، آنها مجبور به انجام پاره‌ای محاسبات و اندازه‌گیری هستند. در این بخش به موضوع اندازه‌گیری کارآیی شبکه نگاهی می‌اندازیم. مباحث ذیل برگرفته از پژوهشهای Mogul (۱۹۹۳) می‌باشد. روال بهبود کارآیی شبکه شامل مراحل زیر است:

۱. **پارامترهای شبکهٔ مورد نظر و کارآیی آن را اندازه‌گیری کنید.**
۲. **سعی کنید وضعیت فعلی شبکه را ارزیابی نمایید.**
۳. **یکی از پارامترها را تغییر بدهید.**

این مراحل آنقدر تکرار می‌شوند تا کارآیی شبکه به حد مطلوب برسد یا روشن شود که آخرین حد کارآیی همین است.

اندازه‌گیریها را می‌توان به روشهای متعدد و در موقعیتهای متفاوت انجام داد (به صورت فیزیکی یا در پشتهٔ پروتکل). یکی از اساسی‌ترین نوع اندازه‌گیری آن است که تایمری را در ابتدای انجام یک عمل روشن کرده و بررسی کنیم که آن عمل چقدر طول می‌کشد. به عنوان مثال دانستن زمانی که طول می‌کشد تا پس از ارسال TPDU، دریافت آن تأیید شود، بسیار اهمیت دارد. اندازه‌گیریهای دیگری را نیز می‌توان به کمک شمارنده‌ها انجام داد (مثلاً تعداد TPDUهایی که از دست رفته‌اند). یکی دیگر از پارامترهایی که دانستن آن اهمیت دارد تعداد بایتهایی است که در یک مدت زمان معین پردازش می‌شوند.

اندازه‌گیری کارآیی و پارامترهای شبکه پیچیدگیهای خاص خود را دارد. در زیر به برخی از آنها اشاره می‌کنیم. در هر گونه تلاش سیستماتیک برای اندازه‌گیری کارآیی شبکه باید موارد ذیل را مد نظر داشته باشید:

مطمئن شوید که تعداد نمونه‌های آماری به قدر کافی زیاد است

فقط به اندازه‌گیری زمان ارسال **یک** TPDU اکتفا نکنید بلکه اندازه‌گیریهای خود را مثلاً یک میلیون بار تکرار کرده و میانگین آن را بدست بیاورید. در اختیار داشتن تعداد بسیار زیاد نمونه‌های آماری، عدم قطعیت مقدار میانگین و

---

[1]. به عبارتی پس از ارسال ۴۰ میلیون بیت بر روی یک خط 1Gbps تازه اولین پیغام ACK باز خواهد گشت. فرستنده باید قادر به ارسال و بافر کردن چنین حجمی از داده باشد. -م

انحراف معیار اندازه‌گیری شده را کاهش خواهد داد. میزان «عدم قطعیت» به کمک فرمولهای استاندارد آماری قابل محاسبه است.

دقت داشته باشید که نمونه‌های آماری به صورت جامع انتخاب شده‌اند

ایده‌آل آنست که مثلاً یک میلیون بار اندازه‌گیری و نمونه‌برداری، در زمانهای مختلف یک روز و در طی هفته تکرار شود تا تأثیر بار سیستم در زمانهای متفاوت بر روی کمیت اندازه‌گیری شده، مشخص گردد. به عنوان مثال اندازه‌گیری ازدحام در لحظه‌ای که ازدحام وجود ندارد، چندان مفید نیست. گاهی اوقات نتایج بدست آمده در بدو امر غیرطبیعی به نظر می‌رسند (مثل ازدحام سنگین در خلال ساعت ۱۰ تا ۲ و عدم ازدحام در حول و حوش ظهر ـوقت ناهار– که کاربران کار خود را رها کرده‌اند!).

وقتی از ساعت نه چندان دقیق استفاده می‌کنید مراقب نتیجهٔ اندازه گیریها باشید

ساعت کامپیوترها بدین نحو کار می‌کنند که در فواصل زمانی معین به یک یا چند شمارنده، یک واحد می‌افزایند. به عنوان مثال تایمر یک میلی‌ثانیه‌ای در هر میلی‌ثانیه، یک واحد به شمارنده اضافه می‌کند. استفاده از چنین تایمری برای اندازه‌گیری رخدادهایی که کمتر از یک میلی‌ثانیه طول می‌کشد امکان‌پذیر است ولی مستلزم دقت زیادیست. (البته بعضی از کامپیوترها ساعت دقیقتری دارند.)

برای محاسبهٔ زمان ارسال یک TPDU، بایستی ساعت سیستم (که مثلاً دقت آن میلی‌ثانیه است) در ابتدای شروع «کُد برنامهٔ واحد انتقال» و همچنین به محض خاتمهٔ کار خوانده شود. اگر زمان واقعی ارسال یک TPDU، ۳۰۰ میکروثانیه باشد، اختلاف بین دو زمان خوانده شده یا صفر است یا یک، که هر دوی آنها اشتباه است. ولیکن اگر اندازه‌گیری زمان برای ارسال یک میلیون TPDU انجام گیرد و زمان کل بر عدد یک میلیون تقسیم شود، میانگین زمان دقتی حدود یک میکروثانیه خواهد داشت.

مطمئن باشید که در خلال آزمایشات هیچ چیز پیش‌بینی نشده‌ای وجود ندارد

انجام اندازه‌گیری بر روی سیستم یک دانشگاه در روزی که مثلاً چندین پروژهٔ آزمایشگاهی عظیم تحت آزمایش است می‌تواند نتیجهٔ کاملاً متفاوتی با اندازه‌گیریهای روز بعد داشته باشد. به دلیل مشابه، اگر چندین پژوهشگر در حین آزمایشهای آماری شما تصمیم به اجرای یک کنفرانس ویدیویی بر روی شبکه بگیرند ممکن است شما را با نتایج گمراه‌کننده‌ای مواجه سازند. بهترین کار آن است که بررسیها را بر روی یک سیستم بیکار انجام بدهید و باری را که می‌خواهید بر روی شبکه یا سیستم بگذارید خودتان ایجاد کنید. البته این راهکار هم ممکن است معضلات خود را داشته باشد. شاید با خود بیندیشید که هیچکس در ساعت ۳ بامداد از شبکه استفاده نمی‌کند در حالی که امکان دارد در همین زمان برنامه‌هایی که بطور خودکار نسخه‌ای پشتیبان از دیسک را بر روی نوار مغناطیسی منتقل می‌کنند در حال کار باشند. مضاف بر این ممکن است ترافیک سنگینی از وب‌سایت شما به نقطه‌ای از جهان (که ساعت محلی آنها متفاوت است) در حال انتقال باشد.

فرآیند ذخیره در حافظهٔ نهان (Caching) می‌تواند صحّت نتایج اندازه‌گیری را به خطر بیندازد

روش بدیهی برای اندازه‌گیری زمان انتقال فایل آن است که یک فایل بزرگ را باز کنید، کلّ آن بخوانید و نهایتاً آنرا ببندید و سپس زمان کلّ را محاسبه نمایید. برای بالا بردن دقت محاسبه، باید این کار چندین بار تکرار شده و میانگین گرفته شود. مشکل اینجاست که امکان دارد، سیستم این فایل را در حافظه نهان ذخیره کرده باشد؛ بنابراین فقط اولین اندازه‌گیری با ترافیک شبکه سر و کار دارد و بقیهٔ اندازه‌گیریها کاملاً غلط هستند چرا که از بافر محلی خوانده می‌شوند. نتیجه‌ای که از چنین آزمایشی بدست می‌آید به کل فاقد اعتبار است. (مگر آن که خواسته باشید

کارآیی حافظهٔ نهان ـ cache ـ را اندازه‌گیری کنید!)

اغلب می‌توانید با سرریز کردن بافر آن را دور بزنید. به عنوان مثال اگر فضای حافظهٔ نهان ۱۰ مگابایت باشد در حلقهٔ آزمایش می‌توانید **دو** فایل ده مگابایتی را باز کنید؛ سپس آنها را پشت سرهم بخوانید و نهایتاً آنها را ببندید تا در هر بار که یکی از فایلها آزمایش می‌شود، میزان استفاده از حافظهٔ نهان (cache) به صفر برسد. با این وجود توصیه می‌شود با احتیاط این کار را انجام دهید مگر آن که کاملاً با الگوریتم caching خود آشنا باشید.

بافرسازی داده‌ها نیز نتیجهٔ مشابهی دارد. مشهور است که یکی از برنامه‌های رایج اندازه‌گیری کارآیی TCP/IP، کارآیی UDP را بیشتر از ظرفیت ممکن خط، گزارش می‌کند!! چرا این اتفاق افتاده است؟ دلیل این است که پس از فراخوانی UDP به محض آن که پیام ارسالی تحویل هستهٔ سیستم عامل می‌شود، آن پیام در انتهای صف ارسال قرار گرفته و کنترل اجرا به برنامهٔ فراخواننده بر می‌گردد. اگر فضای بافر به مقدار کافی وجود داشته باشد حتی هزار بار فراخوانی UDP به معنای ارسال آنها نخواهد بود. بیشتر آنها هنوز در اختیار هستهٔ سیستم عامل هستند ولی ابزار محاسبهٔ کارآیی فکر می‌کند که واقعاً ارسال شده‌اند.

دقیقاً بدانید که چه چیزی را اندازه‌گیری می‌کنید

وقتی زمان خوانده شدن یک فایل راه دور را اندازه می‌گیرید، اندازه‌گیری شما به شبکه، سیستم عامل دوطرف (سرویس‌دهنده و مشتری)، سخت‌افزار واسط شبکه، برنامه‌های راه‌انداز آنها (drivers) و عوامل دیگر بستگی دارد. در صورتی که اندازه‌گیریها بدقت انجام شده باشد، زمان انتقال فایل براساس پیکربندی فعلی بدست می‌آید. اگر هدف نهایی شما تنظیم همین پیکربندی است این اندازه‌گیری مفید خواهد بود ولیکن اگر همین اندازه‌گیری را بر روی سه سیستم انجام می‌دهید تا براساس آن یک کارت شبکهٔ مناسب برای خرید انتخاب کنید، نتایج بدست آمده می‌تواند کاملاً فاقد اعتبار باشد چرا که مثلاً برنامهٔ راه‌انداز کارت شبکه نامناسب بوده و فقط از ده درصد کارآیی کارت شبکه، بهره‌برداری کرده است!

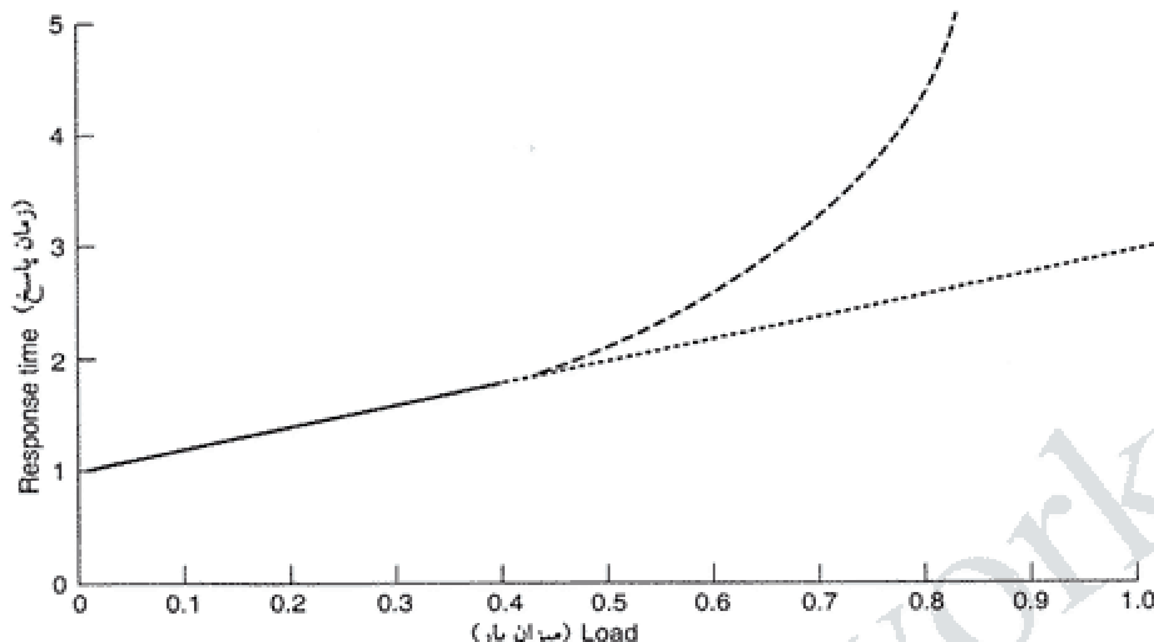
در خصوص نتایج حاصل از برون‌یابی دقت (Extrapolating) داشته باشید

فرض کنید که اندازه‌گیری پارامتری مثل زمان پاسخ را با ایجاد بار شبیه سازی شده از صفر (بیکار) تا ۰/۴ (۴۰ درصد کل ظرفیت) اندازه‌گیری کرده‌اید و یک نمودار شبیه به خط توپر در شکل ۶-۴۲ بدست آورده‌اید. براساس برون‌یابی خطی، نتیجه‌ای اشتباه مثل خط نقطه‌چین بدست می‌آید. در حالی که نتایج آزمایشاتی که با صف سر و کار دارند از عاملی به صورت $1/(1-\rho)$ تأثیر می‌پذیرند که در آن ρ، پارامتر بار (Load) است لذا مقادیر واقعی شبیه به نمودار خط‌چین هستند که شیب بیشتری نسبت به نمودار خطی دارد.

۶-۶-۳ طراحی سیستم برای کارآیی بهتر

اندازه‌گیری و بهره‌برداری از نتایج محاسبات اغلب می‌تواند کارآیی را تا حد قابل توجهی افزایش بدهد ولیکن نمی‌تواند از همان ابتدا جایگزین طراحی خوب شود. کارآیی یک شبکه با طراحی ضعیف را فقط تا حدی می‌توان بهبود داد. پس از آن باید شبکه را از نو طراحی کرد.

در این بخش قواعدی را برای طراحی ارائه می‌دهیم که مبتنی بر تجارب گسترده از شبکه‌های متعدد است. این قواعد مرتبط با طراحی سیستم هستند نه طراحی شبکه، زیرا نرم‌افزار و سیستم عامل، اغلب نقش مهمتری در مقایسه با مسیریابها و کارتهای واسط شبکه در کارآیی ایفاء می‌کنند. بیشتر نظریاتی که ارائه می‌شود برای طراحان شبکه یک دانش پایهٔ عادی و حاصل تجربیاتی است که نسل به نسل منتقل شده و تکامل یافته است. این نظریات برای اولین بار توسط Mogul (۱۹۹۳) به صورت مدون ارائه شد که ما نیز از وی پیروی می‌کنیم. در این زمینه مرجع (Metcalfe, 1993) نیز برای مطالعه مناسب است.



شکل ۶-۴۲. «زمان پاسخ» بعنوان تابعی از «بار».

قانون شمارهٔ ۱: سرعت CPU از سرعت شبکه مهمتر است

تجارب طولانی مدت نشان داده که تقریباً در تمام شبکه‌ها، سربار سیستم عامل و پشتهٔ پروتکلی از زمان واقعی استفاده از کانال بیشتر است.[1] به عنوان مثال از دیدگاه تئوری زمان لازم برای یک عمل RPC (یعنی فراخوانی پروسیجر از راه دور) بر روی شبکهٔ اترنت حداقل ۱۰۲ میکروثانیه است. (شامل انتقال یک تقاضای ۶۴ بایتی و پاسخ ۶۴ بایتی). در عمل غلبه کردن بر سربار تحمیلی توسط نرم‌افزار و رساندن زمان RPC بـه هـمین حـدود موفقیت بزرگی محسوب می‌شود.

به دلیل مشابه، بزرگترین مشکل کار کردن در سرعت 1Gbps ، استخراج بیتها از بافر کاربر و انتقال آن بر روی فیبرنوری و همچنین در اختیار داشتن پردازنده‌ای است که در سمت مقابل بتواند با سرعت کافی آنها را دریافت کند. کوتاه سخن آن که اگر سرعت CPU را دو برابر کنید اغلب توان خروجی نیز به نزدیکی دو برابر می‌رسد. دو برابر کردن ظرفیت شبکه معمولاً نتیجه‌ای در پی ندارد چرا که گلوگاه اصلی در ماشینهای میزبان است.

قانون شمارهٔ ۲: کاهش تعداد بسته‌ها برای کاهش سربار نرم‌افزار

پردازش یک TPDU به میزان معینی سربار به ازای هر TPDU (مثلاً برای پردازش سرآیند) و همچنین به میزان معینی سربار به ازای هر بایت (مثلاً برای محاسبهٔ کد کشف خطای Checksum) به CPU تحمیل می‌کند.[2] وقتی یک میلیون بایت ارسال می‌شود سربار تحمیلی آن به ازای این تعداد بایت ثابت است و به اندازهٔ TPDUها بستگی ندارد. ولیکن سرباری که بدلیل استفاده از TPDUهای ۱۲۸ بایتی تحمیل می‌شود ۳۲ برابر نسبت به زمانی که از TPDUهای ۲کیلوبایتی استفاده می‌گردد، بیشتر خواهد بود.

مضاف بر سربار TPDU در لایهٔ انتقال، در لایه‌های زیرین هم سربار قابل توجهی تحمیل می‌شود. هر بستهٔ دریافتی یک «وقفه» (interrupt) ایجاد می‌کند. در پردازنده‌های مدرن مبتنی بر معماری Pipeline، بروز وقفه (۱)

---

۱. بعبارتی تأخیر تحمیلی توسط نرم‌افزار می‌تواند بیشتر از تأخیر ناشی از انتقال باشد. -م

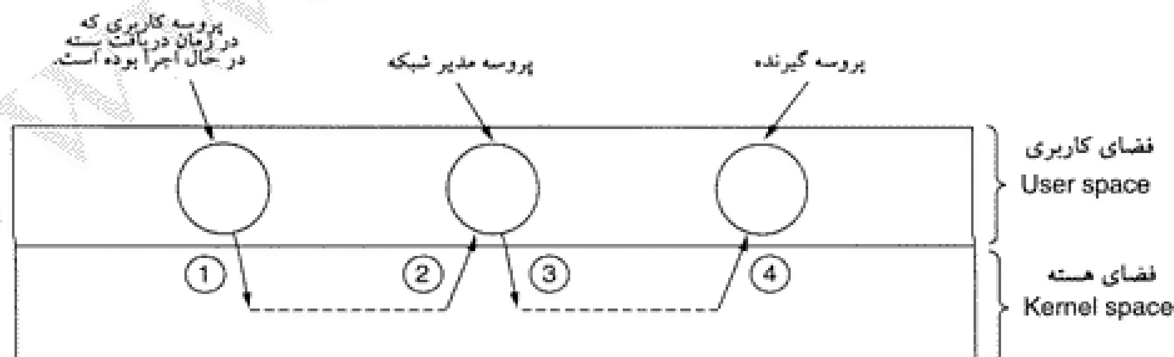
۲. Overhead per TPDU / Overhead per Byte

«خط لولهٔ پردازنده» را قطع می‌کند، (۲) حافظهٔ نهان (cache) پردازنده را درهم می‌ریزد، (۳) نیاز به تغییر روال کار مدیریت حافظه (یعنی عمل تعویض فضای کاری حافظه) دارد (۴) CPU را وادار به ذخیره‌سازی تعدادی رجیستر می‌کند. کاهش تعداد TPDUهای ارسالی با ضریب n ، تعداد وقفه‌ها و میزان سربار بسته‌ها را با ضریب n کاهش خواهد داد.

این واقعیت بیانگر آنست که برای کاهش وقفه‌ها در طرف مقابل، باید قبل از ارسال، ابتدا مقدار قابل توجهی داده، جمع‌آوری شود. الگوریتم Nagle و راه حل Clark که در بخشهای قبلی تشریح شدند بدین منظور مفید هستند.

قانون شمارهٔ ۳ : کاهش تعداد «تعویض متن» (Centext Switch)

انجام عملیات «تعویض متن» (مثلاً از حالت هسته به حالت کاربری) از لحاظ سربار تحمیلی مهلک است! این کار تمام ویژگیهای بد وقفه‌ها را دارد و بدترین آنها این است که دنباله‌ای طولانی از محتوای اولیهٔ حافظهٔ نهان CPU از دست می‌رود. داشتن یک پروسیجر کتابخانه‌ای که داده‌های ارسالی را تا رسیدن به حجم قابل توجه بافر کند، به کاهش دفعات «تعویض متن» کمک خواهد نمود. همینطور در سمت گیرنده، TPDUهای کوچک دریافتی باید جمع‌آوری شده و بطور یکجا تحویل کاربر گردد تا دفعات «تعویض متن» کاهش یابد.
در بهترین حالت، یک بستهٔ ورودی موجب می‌شود که سیستم عامل مجبور به «تعویض متن» از حالت کاربر به حالت هسته[1] باشد. سپس باید از هسته به پروسهٔ دریافت‌کنندهٔ این بستهٔ ورودی تغییر حالت بدهد. متأسفانه در بسیاری از سیستمهای عامل به دفعات بیشتری «تعویض متن» نیاز است. به عنوان مثال، اگر مدیر شبکه پروسهٔ خاصی را در فضای کاربری[2] اجرا کرده باشد ، دریافت یک بسته موجب می‌شود که سیستم عامل ابتدا از کاربر فعلی به هسته، تغییر وضعیت (و تعویض متن) بدهد، سپس تغییر وضعیتی دیگر از هسته به پروسهٔ مدیر شبکه، سپس تغییری دیگر از پروسهٔ مدیر شبکه به هسته و نهایتاً از هسته به پروسهٔ تحویل گیرندهٔ بسته. این توالی در شکل ۶-۴۳ نشان داده شده است. تمام این عملیات تعویض متن که به ازای ورود هر بسته انجام می‌شود، زمان CPU را تا حد بسیار زیادی هدر خواهد داد و تأثیر مخربی بر روی کارآیی شبکه خواهد گذاشت.



شکل ۶-۴۳. چهار بار تعویض متن برای رسیدن یک بسته به پروسهٔ مدیر شبکه در فضای کاربری.

قانون شمارهٔ ٤ : حداقل کردن دفعات کپی برداری

تعدّد کپی برداری حتی از تعدّد دفعات تعویض متن هم بدتر است. این موضوع که بخواهیم یک بستهٔ ورودی را قبل از تحویل محتوای درونی آن به پروسهٔ نهایی، سه یا چهار بار کپی کنیم، امری نامعمول و عجیب نیست. معمولاً

---

۱. User mode to Kernel mode Context Switch
۲. User Space

پس از دریافت یک بسته توسط کارت واسط شبکه و ذخیرهٔ موقت آن در یک بافر سخت‌افزاری ویژه، بسته به درون بافر هستهٔ سیستم عامل کپی می‌شود. از آنجا نیز به بافر لایهٔ شبکه و از آنجا به بافر لایهٔ انتقال و نهایتاً به بافر پروسهٔ کاربردی گیرنده منتقل می‌شود.

یک سیستم عامل هوشمند قادر به کپی یک کلمه در آن واحد است ولیکن ممکن است برای کپی یک کلمه حتی به ۵ دستورالعمل نیاز باشد. (شامل بار کردن کلمه در رجیستر، ذخیرهٔ آن در موقعیت جدید، اضافه کردن به رجیستر شاخص ـindex register، تست برای تشخیص پایان داده‌ها و انشعاب شرطی) سه بار کپی کردن هر بسته در شرایطی که انتقال هر کلمهٔ ۳۲ بیتی با ۵ دستورالعمل انجام می‌شود به ۴÷۱۵ دستورالعمل یعنی حدوداً چهار دستورالعمل به ازای هر بایت، نیاز خواهد داشت. در یک CPU با سرعت 500MIPS ، هر دستورالعمل ۲ نانوثانیه زمان می‌برد لذا سه بار انتقال هر بایت حدوداً ۸ نانوثانیه طول می‌کشد؛ یعنی حدود ۱ نانو ثانیه به ازای هر بیت؛ بدین ترتیب چنین پردازنده‌ای قادر به پردازش 1-Gbps است. وقتی سربار ناشی از پردازش سرآیند، مدیریت وقفه، عملیات تعویض متن (Context Switching) را نیز در نظر بگیریم **شاید** بتوان به سرعت پردازش 500Mbps رسید، در حالی که پردازش اصلی داده‌ها را نیز به حساب نیاورده‌ایم. بدیهی است که پردازش داده‌های شبکهٔ اترنت 10-Gbps که با تمام سرعت ارسال می‌کند، بدین نحو میسر نخواهد بود.

در حقیقت، شاید با CPU فوق، حتی نتوان از عهدهٔ پردازش داده‌های یک خط 500Mbps که با سرعت تمام در جریان است، برآمد. از طرفی در محاسبات فوق، سرعت ماشین را 500Mbps فرض کرده‌ایم یعنی می‌تواند پانصد میلیون دستورالعمل را در هر ثانیه اجرا کند. در دنیای واقعی، ماشینها فقط زمانی می‌توانند با چنین سرعتی کار کنند که به حافظه رجوع نداشته باشند. عملیات حافظه عموماً ده بار کندتر از دستورالعملهای رجیستر به رجیستر اجرا می‌شوند. (به عبارتی ۲۰ نانو ثانیه برای هر دستورالعمل حافظه). اگر بیست درصد از دستورات نیازمند رجوع به حافظه باشند (به عبارتی به بیرون از حافظهٔ نهان رجوع کنند) که در خصوص ورود یک بسته کاملاً محتمل است، متوسط زمان اجرای دستورالعملها ۵/۶ نانو ثانیه خواهد بود. (0.8×2+0.2 ×20) اگر برای پردازش هر بایت چهار دستورالعمل اجرا شود، به ۲۲/۴ نانو ثانیه برای پردازش هر بایت یا ۲/۸ نانو ثانیه برای هر بیت نیاز خواهیم داشت که در این صورت ظرفیت پردازش به 357Mbps کاهش می‌یابد. اگر پنجاه درصد سربار اضافی را در محاسبات خود وارد کنیم، این مقدار به 178Mbps می‌رسد. در اینجا سخت‌افزار نمی‌تواند کمک بیشتری بکند. مشکل اینجاست که تعداد کپی‌هایی که در سیستم عامل انجام می‌شود زیاد است.

قانون شمارهٔ ۵ : می‌توانید پهنای باند بیشتری بخرید ولی تأخیر پایین‌تر خریدنی نیست

سه قانون بعدی به جای آن که به پردازش پروتکل بپردازند در خصوص مسائل ارتباطی شبکه بحث می‌کنند. قانون اول بیان می‌کند که اگر پهنای باند بیشتری خواستید قادر به تهیهٔ آن خواهید بود. قرار دادن یک رشته فیبرنوری دیگر در کنار رشتهٔ قبلی پهنای باند شما را دو برابر می‌کند ولیکن تأخیر را کاهش نخواهد داد. کم کردن تأخیر مستلزم بهبود نرم‌افزار پروتکل، سیستم عامل، یا کارت شبکه است. حتی اگر به تمام این بهبودها نائل شوید ، تأخیر شما هرگز از تأخیر انتقال کمتر نخواهد شد. [بطور ذاتی هر کیلومتر سیم ۵ میکروثانیه و هر کیلومتر فیبرنوری ۳/۳ میکروثانیه تأخیر دارد و این تأخیر اجتناب‌ناپذیر است. -م]

قانون شمارهٔ ۶: پیشگیری از بروز ازدحام بهتر از رفع آن است

قطعاً این مثل قدیمی که: «پیشگیری به مراتب بهتر از درمان است» برای مسئله ازدحام در شبکه نیز صادق است. وقتی شبکه‌ای با ازدحام مواجه می‌شود بسته‌هایی از بین می‌روند، پهنای باند تلف می‌شود، تأخیرهای نامعقول پدید می‌آید و تبعاتی از این قبیل. بیرون آمدن و رفع ازدحام دشوار و زمان‌بر است. بهتر آن است که نگذاریم

ازدحام رخ بدهد. پیشگیری از ازدحام همانند تزریق واکسن DTP است: اندکی دردسر دارد ولی از مشکلات بزرگ پیشگیری می‌کند.

قانون شمارهٔ ۷: اجتناب از انقضای مهلت

وجود تایمرها در هر شبکه‌ای قطعاً لازم است ولی فقط باید در موارد ضروری از تایمر استفاده کرد و مهلت انقضای زمان تایمرها بایستی حداقل (بهینه) باشند. وقتی تایمری به صفر می‌رسد، عموماً عملیاتی از نو تکرار خواهد شد. اگر تکرار این عملیات واقعاً لازم نباشد منابع سیستم بیهوده هدر می‌رود.

برای اجتناب از انجام عملیات زائد باید زمان تایمرها به دقت تنظیم شده و اندکی با مقدار لازم فاصله داشته باشد، زیرا محافظه‌کاری زیاد و تنظیم مقادیر بزرگ در تایمرها باعث بروز تأخیرات نامعقول در هنگام از دست رفتن TPDU می‌شود. از طرفی اگر تایمر در زمانی که واقعاً موعد آن نیست به صفر برسد، زمان CPU صرف عملیات بیهوده می‌شود، پهنای باند به هدر می‌رود و بی‌هیچ دلیل بار زیادی بر روی دهها مسیریاب تحمیل می‌کند.

۶-۶-۴ پردازش سریع TPDU

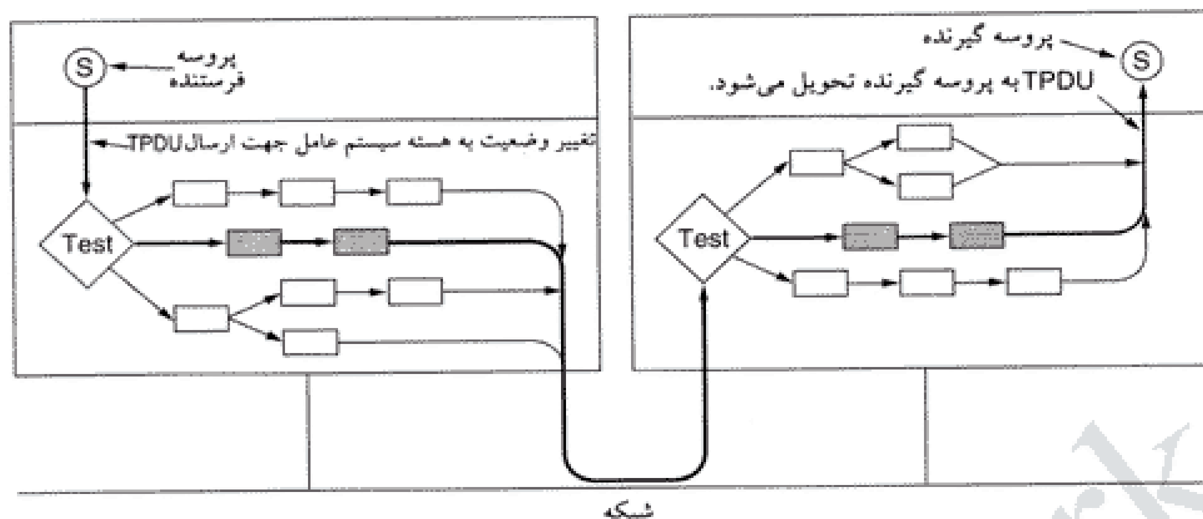
نتیجه‌گیری منطقی از قضایایی که در بالا بدان پرداختیم آنست که مانع اصلی در ایجاد شبکه‌های سریع، نرم‌افزار پروتکل آنهاست. در این بخش روشهایی را مرور می‌کنیم که به این نرم‌افزار سرعت می‌بخشند. برای کسب آگاهی بیشتر به مراجع (Clark et al., 1989; and Chase et al., 2001) مراجعه نمایید.

سربار پردازش TPDU دو عامل دارد: سربار تحمیلی به ازای هر TPDU و سربار تحمیلی به ازای هر بایت. هر دوی آنها را باید کاهش داد. عامل کلیدی در پردازش سریع TPDU آن است که TPDUهای معمولی (حاوی داده) را از مابقی جدا کرده و آنها را به صورت ویژه پردازش نماییم. اگرچه به دنباله‌ای از TPDUهای خاص نیاز است تا بتوان به حالت ESTABLISHED وارد شد ولیکن از اینجا به بعد پردازش TPDU سرراست و ساده خواهد بود، تا وقتی که یکی از طرفین اقدام به قطع اتصال نماید.

اجازه بدهید فرض را بر آن بگذاریم که طرف فرستنده اقدام به برقراری اتصال کرده و اکنون در حالت ESTABLISHED قرار دارد و می‌خواهد داده‌هایی را بفرستد. برای سادگی بحث فرض می‌کنیم که «واحد انتقال» (Transport Entity) درون هستهٔ سیستم عامل قرار دارد؛ اگرچه ایده‌هایی که مطرح می‌شوند برای زمانی که «واحد انتقال» یک پروسه در فضای کاربری است یا حتی به صورت توابع کتابخانه‌ای درون پروسهٔ فرستنده جای گرفته، نیز صادق هستند. در شکل ۶-۴۴، پروسهٔ فرستندهٔ داده، به هستهٔ سیستم عامل رجوع می‌کند تا عملیات SEND (ارسال) انجام شود. اولین کاری که واحد انتقال انجام می‌دهد بررسی و تشخیص حالت معمولی است یعنی باید: (۱) اتصال در حالت ESTABLISHED قرار داشته باشد، (۲) هیچیک از طرفین سعی در بستن اتصال نکرده باشند، (۳) TPDU کامل و معتبر باشد و (۴) در سمت گیرنده فضای پنجره به اندازهٔ کافی موجود باشد. اگر تمام شرایط فوق برآورده شود به آزمایش بیشتری نیاز نیست و آن TPDU می‌تواند در مسیر **پردازش سریع** قرار بگیرد. طبعاً اکثر بسته‌های TPDU از همین مسیر می‌گذرند.

در حالت عادی سرآیند TPDUهای متوالی و حاوی داده، تقریباً مشابه هم هستند.[۱] برای بهره‌برداری مفید از این ویژگی، «نمونهٔ سرآیند» (Header Prototype) درون واحد انتقال کپی می‌شود. در ابتدای مسیر پردازش سریع، سرآیند TPDU با حداکثر سرعت و کلمه به کلمه درون بافر جدید کپی می‌گردد. فیلدهایی که از یک TPDU به TPDU دیگر تغییر می‌کنند درون بافر رونویسی می‌شوند. عموماً مقدار فیلدهایی را که در هر TPDU

۱. از لحاظ مقدار «تقریباً» مشابهند وگرنه از لحاظ ساختار سرآیند یقیناً مثل هم هستند. -م



شکل ۶-۴۴. مسیر پردازش سریع بسته‌ها از فرستنده به گیرنده با خطّ تیره‌تر مشخص شده است. مراحل پردازش نیز بصورت خاکستری نشان داده شده است.

تغییر می‌کنند (همانند فیلد شمارهٔ ترتیب و شمارهٔ Ack)، به سادگی می‌توان از «متغیرهای حالت»، موجود در واحد انتقال بدست آورد. سپس یک اشاره‌گر به سرآیند کامل TPDU، به همراه اشاره‌گر دومی به داده‌های کاربر، تحویل لایهٔ شبکه می‌شود. در لایهٔ شبکه نیز استراتژی مشابهی دنبال می‌شود. (استراتژی لایهٔ شبکه در شکل ۶-۴۴ نشان داده نشده است.) در آخر، لایهٔ شبکه بستهٔ حاصله را جهت انتقال به لایهٔ پیوند داده تحویل می‌دهد.[۱]

برای آن که ببینیم این روال در عمل چگونه کار می‌کند، TCP/IP را در نظر می‌گیریم. در شکل ۶-۴۵-الف سرآیند TCP نشان داده شده است. فیلدهایی که در TPDUهای متوالی مشابه هستند (یعنی مقدار آنها تغییر نمی‌کند) به صورت خاکستری نشان داده شده‌اند. تمام کاری که «واحد انتقال» انجام می‌دهد آن است که (۱) این پنج کلمه را از درون «نمونهٔ سرآیند» (Header Prototype) به بافر خروجی منتقل کند؛ (۲) شمارهٔ ترتیب بعدی را در محل مربوطه قرار بدهد (از طریق انتقال یک کلمه در حافظه)، (۳) کد جمع کنترلی (Checksum) را محاسبه کند و (۴) به شمارهٔ ترتیب در حافظه اضافه نماید. سپس سرآیند تنظیم شده و داده‌ها تحویل پروسیجر IP می‌شود تا بطور طبیعی ارسال گردد. نرم‌افزار IP نیز «نمونهٔ سرآیند» پنج کلمه‌ای خود را (شکل ۶-۴۵-ب) به درون بافر کپی می‌کند، فیلد Identification را مقداردهی و کد جمع کنترلی آن را محاسبه می‌نماید. اکنون بسته، آماده ارسال است.

حال ببینیم، مسیر پردازش سریع در طرف گیرنده از شکل ۶-۴۴ چگونه کار می‌کند. در مرحلهٔ ۱ ابتدا «**رکورد اتصال**» مربوط به TPDU ورودی پیدا می‌شود. در نرم‌افزار TCP، رکورد اتصال می‌تواند در یک «جدول درهم‌سازی» (Hash Table) ذخیره شده و کلید آن برحسب تابعی ساده از آدرس IP و دو شمارهٔ پورت محاسبه گردد. به محض پیدا شدن موقعیت رکوردِ متناظر با بستهٔ ورودی، ابتدا آدرسهای IP و شماره‌های پورت آن با این رکورد مقایسه می‌شود تا صحت رکورد پیدا شده تأیید گردد.

---

۱. اگر بخواهیم عملکرد «مسیر پردازش سریع» را به زبان ساده بیان کنیم بدین نحو عمل می‌کند که اگر TPDU از نوع معمولی و حاوی داده باشد، برای آن از قبل یک سرآیند ایجاد می‌شود که خوشبختانه برای هر TPDU جدید فقط تعداد کمی از فیلدهای آن تغییر می‌کند. بلافاصله پس از تنظیم سرآیند هر TPDU، اشاره‌گر محل شروع این سرآیند و اشاره‌گر محل شروع داده‌های کاربر، تحویل لایهٔ شبکه می‌شود. بنابراین بجای انتقال و کپی یک بستهٔ کامل، اشاره‌گرها بین دو لایه ردّ و بدل می‌شوند. -م

Source port			Destination port				
Sequence number							
Acknowledgement number							
Len	Unused		Window size				
Checksum			Urgent pointer				

(الف)

VER.	IHL	TOS	Total length
Identification			Fragment offset
TTL		Protocol	Header checksum
Source address			
Destination address			

(ب)

شکل ۶-۴۵. (الف) سرآیند TCP (ب) سرآیند IP. در هر دو سرآیند، فیلدهای خاکستری بدون هیچ تغییری از «نمونهٔ سرآیند» (Header Prototype) موجود گرفته می‌شود.

یک بهینه‌سازی که می‌تواند سرعت جستجوی «رکورد اتصال» را افزایش بدهد آن است که همیشه اشاره‌گر آخرین رکوردی که از آن استفاده شده، نگهداری گردد و جستجو از آن نقطه آغاز شود. کلارک و همکارانش (Clark et al. 1989) این بهینه سازی را آزمایش و مشاهده کردند که احتمال موفقیت سریع در جستجو به بیش از ۹۰ درصد افزایش می‌یابد. روشهای جستجوی دیگر در مرجع (McKenney and Dove; 1992) تشریح شده است.

سپس TPDU بررسی می‌شود تا مشخص گردد که آیا از نوع معمولی است یعنی: اتصال در حالت ESTABLISHED قرار داشته باشد، هیچیک از طرفین تقاضای خاتمهٔ اتصال نداده باشند، TPDU یک بستهٔ کامل و معتبر باشد، هیچیک از بیتهای پرچم (Flag Bits) در TPDU، تنظیم نشده باشد و شمارهٔ ترتیب TPDU دریافتی همان شمارهٔ مورد نظر باشد. این بررسی‌ها مستلزم اجرای چندین دستورالعمل ماشین است. اگر این شرایط برآورده شود. یک «پروسیجر پردازش سریع» برای آن TPDU فراخوانی می‌شود.

«مسیر پردازش سریع» رکورد اتصال را بهنگام‌سازی کرده و داده‌ها را برای پروسهٔ کاربر کپی می‌نماید. در حین کپی، کد جمع کنترلی (checksum) داده‌ها را محاسبه می‌کند تا نیاز به یک گذر اضافی برای این عمل نیاز نباشد. اگر کد جمع کنترلی صحیح بود، رکورد اتصال بهنگام سازی شده و پیغام Ack بازگردانده می‌شود. الگوی کلی تست سریع سرآیند برای تشخیص آن که آیا سرآیند TPDU، سرآیند مورد انتظار هست یا خیر توسط پروسیجری به نام **«پیشگویی سرآیند»** (Header Prediction) انجام می‌شود و در بسیاری از پیاده‌سازیهای عملی TCP از آن استفاده شده است. هر گاه این بهینه‌سازی و بهینه‌سازیهای دیگری که در این فصل تشریح کردیم در کنار هم مورد استفاده قرار بگیرد می‌توان به ۹۰ درصد سرعت کپی داده‌ها از حافظه به حافظه رسید. (با فرض آن که شبکه به قدر کافی سریع است.)

دو مورد دیگر که به کمک آنها می‌توان بهبود چشمگیری در کارآیی بوجود آورد مدیریت بافر و مدیریت تایمرهاست. موضوع مرتبط با مدیریت بافر، اجتناب از کپی‌برداریهای بی‌مورد است. به این موضوع در بالا اشاره کردیم. مدیریت تایمر اهمیت بیشتری دارد چرا که تقریباً اغلب آنها (اگر بدرستی تنظیم شده باشند) منقضی نمی‌شوند: آنها بدین منظور تنظیم و راه‌اندازی می‌شوند که اگر یک TPDU از بین رفت از نو ارسال شود در حالی که اغلب TPDUها به درستی تحویل و پیغام اعلام وصول آنها نیز به سلامت بر می‌گردند. بنابراین مدیریت بهینهٔ تایمرها بسیار اهمیت دارد.

یک ساختار رایج برای مدیریت تایمرها آن است که از یک «لیست پیوندی» (Linked List) متشکل از

رخدادهای تایمر که برحسب زمان انقضای آنها مرتب شده، استفاده گردد.[1] اولین عنصر این لیست پیوندی حاوی یک شمارنده است که تعیین می‌کند تا انقضای مهلت آن، چند تیک ساعت باقی مانده است. هریک از عناصر متوالی این لیست پیوندی شمارنده‌ای دارند که مشخص می‌کند در مقایسه با عنصر قبلی در لیست، چند تیک ساعت بعدتر منقضی می‌شوند. بنابراین اگر سه تایمر باید به ترتیب پس از ۳، ۱۰ و ۱۲ تیک منقضی شوند مقادیر شمارنده‌های آنها در لیست پیوندی به ترتیب ۳ و ۷ و ۲ خواهد بود.

در هر تیک ساعت، شمارندهٔ اولین عنصر لیست پیوندی یک واحد کاهش می‌یابد. هر گاه این شمارنده به صفر برسد، رخداد مربوطه پردازش شده و با حذف آن از لیست، عنصر بعدی در جلوی این لیست قرار می‌گیرد. بدون آن که نیازی به تغییر در مقدار شمارندهٔ این عنصر باشد. در این روش، حذف و اضافهٔ یک تایمر به لیست پیوندی، عملیاتی پر هزینه است و زمان انجام این دو عمل متناسب با طول لیست پیوندی می‌باشد.

اگر مقدار حداکثر تایمرها محدود و از قبل معلوم باشد می‌توان از روش کارآمدتری استفاده کرد. در این روش از آرایه‌ای به نام «**چرخ زمان‌سنجی**» بهره گرفته می‌شود. این آرایه در شکل ۶-۴۶ نشان داده شده است. هر «**برش**» (Slot) متناظر با یک تیک ساعت است. در این شکل زمان فعلی، T=4 فرض شده است. تایمرها به گونه‌ای برنامه‌ریزی شده‌اند که نسبت به زمان فعلی، پس از ۳، ۱۰ و ۱۲ تیک ساعت منقضی شوند. اگر در همین لحظه به تایمری احتیاج شد که قرار است پس از ۷ تیک ساعت منقضی شود، یک درایه (Entry) برای آن در برش ۱۱ (Solt 11) ایجاد می‌شود. به روش مشابه اگر نیاز شد که تایمر T+1 حذف شود، لیستی که شروع آن در برش ۱۴ مشخص شده جستجو و درایهٔ مورد نظر حذف می‌شود. دقت کنید که آرایهٔ شکل ۶-۴۶ نمی‌تواند به غیر از تایمرهایی را که زیر T+15 هستند ایجاد کند. [به عبارتی حداکثر زمان قابل تنظیم در تایمرها معادل ۱۵ تیک ثانیه است.]



شکل ۶-۴۶. یک «چرخ زمان‌سنجی» (Timing Wheel).

۱. یعنی هر یک از عناصر این لیست پیوندی مشخص می‌کنند که چقدر بعد چه اتفاقی باید بیفتد. -م

وقتی ساعت تیک می‌زند، اشاره‌گر زمان فعلی به اندازهٔ یک برش در آرایه جلو می‌رود. (به صورت چرخشی) اگر درایه‌ای که اکنون به آن اشاره می‌شود غیرصفر باشد، تمام تایمرهای تعریف شده در آن پردازش می‌شوند.[۱] گونه‌های بی‌شماری از روش فوق در مرجع (Varghese, Lauck, 1987) بحث شده است.

۶-۶-۵ پروتکلهایی برای شبکه‌های گیگابیتی

شبکه‌های گیگابیتی در ابتدای دههٔ نود در صحنه ظاهر شدند. عموم افراد سعی کردند که از همان پروتکلهای قدیمی بر روی شبکهٔ جدید استفاده کنند ولی چیزی نگذشت که مشکلات خود را نشان دادند. در این بخش به برخی از این مشکلات و راهکارهایی که پروتکلهای جدید برای حرکت به سمت شبکه‌های سریعتر برگزیده‌اند، خواهیم پرداخت.

اولین مشکل آن است که اکثر پروتکلهای فعلی، از شماره ترتیب ۳۲ بیتی برای بسته‌ها، استفاده کرده‌اند. وقتی اینترنت شروع به کار کرد، خطوط مابین مسیریابها، غالباً خطوط اجاره‌ای 56Kbps بودند و اگر یک ماشین با تمام سرعت اقدام به ارسال می‌کرد بیش از یک هفته طول می‌کشید تا شمارهٔ ترتیب ۳۲ بیتی به مقدار قبلی آن برگردد. برای طراحان TCP، عدد 2^{32} تقریب بسیار خوبی از بی‌نهایت تلقی می‌شد زیرا خطر آن که بسته‌ای برای یک هفته در زیرشبکه سرگردان باشد و سپس به صورت تکراری دریافت شود عملاً وجود نداشت. در شبکهٔ اترنت 10Mbps، زمان تکرار اعداد ۳۲ بیتی به ۵۷ دقیقه کاهش یافت ولی هنوز قابل تحمل و مدیریت‌پذیر بود. در اترنت یک گیگابیت بر ثانیه، زمان تکرار اعداد به ۳۴ ثانیه رسید که متأسفانه کمتر از طول عمر حداکثر هر بسته یعنی ۱۲۰ ثانیه است. در اینجا 2^{32} تقریب خوبی از مقدار بی‌نهایت نیست و ممکن است در حالی که هنوز بسته‌های قدیمی در زیرشبکه موجودند بسته‌هایی با شمارهٔ تکراری تولید شده و گیرنده احتمالاً به اشتباه بیفتد. اگرچه در RFC 1323 راهی موقت برای حل این مشکل پیشنهاد شده است.

مشکل از آنجا ناشی می‌شود که طراحان پروتکل به سادگی فرض کرده‌اند زمانی که طول می‌کشد تا از کل فضای ۳۲ بیتی شماره ترتیب استفاده شود از حداکثر طول عمر بسته بیشتر است. در نتیجه (با این فرض) لازم نبود کسی در خصوص وجود بسته‌هایی با شماره‌های تکراری (که در اثر برگشت مقادیر ۳۲ بیتی به مقدار قبلی پیش می‌آید) نگران باشد. در سرعتهای گیگابیتی این فرض نانوشته با شکست مواجه می‌شود.

مشکل دوم آن است که سرعت مخابرهٔ داده‌ها از سرعت پردازش کامپیوترها پیشی گرفته است. (قابل توجه مهندسین کامپیوتر: به میدان رفته و مهندسین مخابرات را شکست بدهید!! ما روی شما حساب می‌کنیم!) در اوائل دههٔ هفتاد میلادی، شبکهٔ ARPANET از خطوط 56Kbps و کامپیوترهایی با سرعتی حدود 1 MIPS بهره می‌گرفت. بسته‌های اطلاعاتی نیز ۱۰۰۸ بیت بودند و بدین ترتیب ARPANET قادر به تحویل حدود ۵۶ بسته در هر ثانیه بود. هر ماشین میزبان در زمانی حدود ۱۸ میلی‌ثانیه که به ازای هر بسته مهلت پردازش داشت، می‌توانست ۱۸۰۰۰ دستورالعمل ماشین را اجرا نماید. البته اختصاص تمام این دستورالعملها به پردازش بسته، کل زمان CPU را مصرف می‌کند ولیکن اگر فقط ۹۰۰۰ دستورالعمل ماشین به پردازش هر بسته اختصاص داده شود، نیمی از توان CPU برای کارهای اصلی باقی می‌ماند.

این اعداد و ارقام را با کامپیوترهای 1000 MIPS که بسته‌های ۱۵۰۰ بایتی را بر روی خطوط گیگابیتی منتقل

---

۱. به زبان ساده، آرایهٔ فوق را در یک دایره با ۱۶ قطاع تجسم کنید که عقربه‌ای روی آن حرکت می‌کند و در هر لحظه در یکی از این قطاعها قرار می‌گیرد. اگر به فرض نیاز به تایمری داشتید که باید ۵ تیک ثانیه بعد منقضی شود نسبت به مکان فعلی عقربه، ۵ قطاع جلوتر، اشاره‌گر آن را قرار می‌دهید. هر بار که عقربه یک قطاع جلو می‌رود تمام کارهایی که اشاره‌گرشان درون آن قطاع قرار دارد، انجام می‌شود. -م

می‌کنند، مقایسه نمایید. در چنین شبکه‌ای نرخ ارسال بیش از ۸۰۰۰۰ بسته در ثانیه است و طبعاً اگر بخواهیم نیمی از توان پردازشی CPU را به بقیهٔ برنامه‌های کاربردی اختصاص بدهیم، بایستی پردازش هر بسته در ۶/۲۵ میکروثانیه تکمیل شود. یک کامپیوتر 1000MIPS در ۶/۲۵ میکروثانیه قادر به اجرای ۶۲۵۰ دستورالعمل است یعنی حدود یک سوم تعداد دستورالعملهایی که ماشینهای ARPANET قادر به اجرای آنها برای هر بسته بودند. مضاف بر این هر دستورالعمل در ماشینهای مدرن RISC ، نسبت به دستورالعمل ماشینهای قدیمی‌تر CISC کار کمتری انجام می‌دهد لذا مشکل حادتر از آن چیزی است که به نظر می‌رسد. نتیجه‌گیری آن است که برای پردازشهایی که در نرم‌افزار پروتکلها انجام می‌شود زمان کمتری در اختیار است فلذا پروتکلها باید ساده‌تر و سریعتر شوند.

مشکل سوم آن است که پروتکل Go Back n بر روی خطوطی که در آنها حاصلضرب پهنای باند در تأخیر بسیار زیاد است، ضعیف عمل می‌کند. به عنوان مثال به یک خط چهار هزار کیلومتری که با سرعت 1-Gbps کار می‌کند، دقت نمایید: زمان تأخیر رفت و برگشت ۴۰ میلی‌ثانیه است و در این زمان فرستنده قادر به ارسال ۵ مگابایت داده می‌باشد؛ طبعاً اگر خطایی گزارش شود مربوط به ۴۰ میلی‌ثانیه قبل بوده است. اگر از پروتکل Go Back n (عقبگرد به اندازهٔ n) استفاده شده باشد، فرستنده نه تنها مجبور است بستهٔ خراب را از نو بفرستد بلکه باید ۵ مگابایت بسته‌های بعد از آن را نیز مجدداً ارسال نماید. به وضوح استفاده از این روش، اتلاف بسیار زیادی در منابع شبکه خواهد بود.

مشکل چهارم آن است که خطوط گیگابیتی تفاوت بنیانی با خطوط مگابیتی دارند: این تفاوت از آنجاست که خطوط طولانی گیگابیتی نسبت به «تأخیر» محدودیت دارند در حالی که دیگری محدودیت «پهنای باند» دارد. در شکل ۶-۴۷ زمانی که طول می‌کشد یک فایل یک مگابیتی از طریق خطی ۴۰۰۰ کیلومتری با نرخهای متفاوت ارسال شود، نشان داده شده است. تا سرعت 1-Mbps، زمان انتقال متأثر از نرخ ارسال بیتهاست. در سرعت 1-Gbps، تأخیر ۴۰ میلی‌ثانیه‌ای رفت‌وبرگشت بر زمان ۱ میلی‌ثانیه‌ای انتقال بیتها بر روی فیبرنوری غالب می‌شود. از اینجا به بعد افزایش پهنای باند تأثیر چندانی در تأخیر انتقال ندارد.

شکل ۶-۴۷ نکات مأیوس‌کننده‌ای برای پروتکلهای شبکه دارد. این شکل بیانگر آن است که پروتکلهای «توقف و انتظار» (Stop & Wait) همانند RPC از لحاظ کارآیی محدودیت ذاتی دارند. این محدودیت به خاطر سرعت نور تحمیل می‌شود و هیچگونه پیشرفت تکنولوژیک در فیزیک نور نمی‌تواند بر این محدودیت فائق آید.

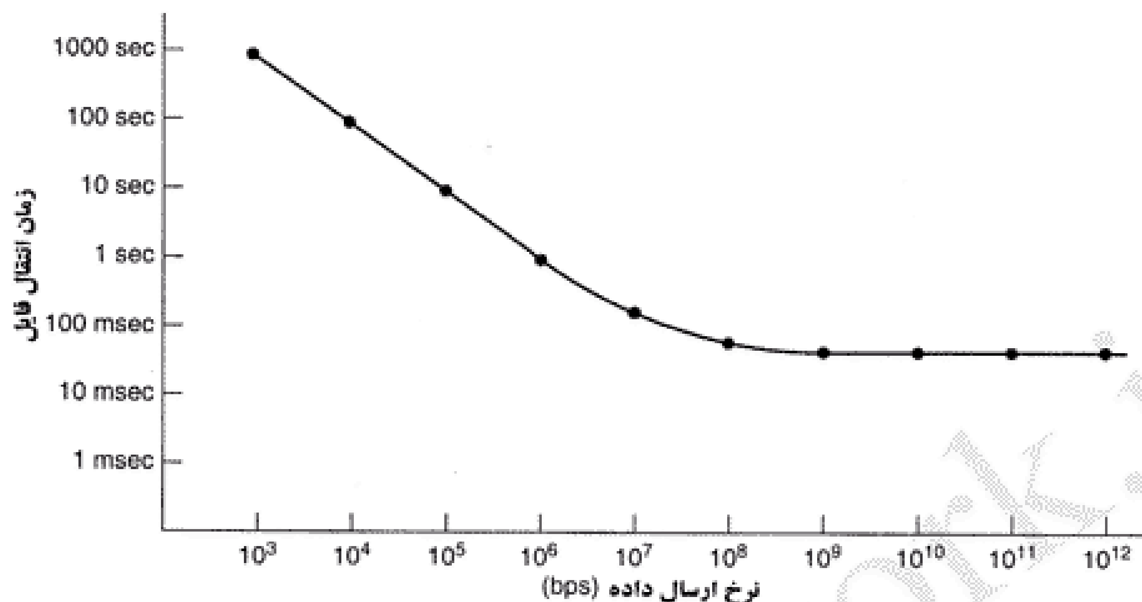
مشکل پنجم که اشاره به آن خالی از لطف نیست ریشه در مسائل تکنولوژیک، پروتکل یا مسائلی بدین شکل ندارد بلکه ناشی از کاربردهای جدید شبکه است. در یک بیان ساده در بسیاری از کاربردهای شبکهٔ گیگابیتی (مثل کاربردهای چند رسانه‌ای) «واریانس زمان دریافت بسته‌ها»، اهمیتی همسنگ با «تأخیر متوسط دریافت» آنها دارد. تحویل آهسته ولی یکنواخت بسته‌ها ارجحتر از تحویل سریع ولی ناگهانی آنهاست.

پس از معرفی مشکلات اجازه بدهید به روشهای حل و فصل آنها بپردازیم. ابتدا تذکراتی کلی ارائه می‌کنیم و سپس به مکانیزمهای پروتکل، ساختار بسته‌ها و نرم‌افزار پروتکل نگاهی خواهیم انداخت.

یک اصل اساسی که تمام طراحان شبکه باید در نظر داشته باشند آن است که:

«طراحی باید مبتنی بر سرعت بالا باشد نه برای بهینه‌سازی پهنای باند»

پروتکلهای قدیمی‌تر اغلب بدان منظور طراحی شده‌اند که تعداد بیتهای ارسالی بر روی کانال به حداقل برسد و بدین منظور از فیلدهای کوچک و ادغام چند فیلد در یک بایت یا کلمه بهره گرفته‌اند. امروزه پهنای باند فراوانی در اختیار است و میزان پردازش در پروتکل به یک مشکل تبدیل شده است، فلذا پروتکلها باید برای به حداقل رساندن حجم پردازش لازم، طراحی شوند. طراحان IPv6 به روشنی این اصل اساسی را درک کرده بودند.



شکل ۶-۴۷. زمان انتقال و تصدیق وصول یک فایل یک مگابیتی از طریق یک خط ۴۰۰۰ کیلومتری.

یک راهکار اغواکننده برای رسیدن به سرعت بالا، ساختن کارتهای واسط شبکه به کمک سخت‌افزار سریع است. [یعنی بخشی از پروتکلهای نرم‌افزاری توسط سخت‌افزار شبکه پیاده‌سازی شود.] مشکل در پیش گرفتن این استراتژی آن است که بدون داشتن یک پروتکل بسیار ساده، کارت سخت‌افزاری به خودی خود تبدیل به یک برد اضافی با یک CPU مستقل و یک برنامهٔ خاص می‌شود. برای آنکه «کمک‌پردازندهٔ شبکه» ارزانتر از پردازندهٔ اصلی تمام شود، اغلب از CPU کُندتر استفاده می‌شود. در نتیجه اگر خود پروتکل ساده نباشد، CPU سریع باید آنقدر منتظر بماند تا CPU دیگر (CPU کُندتر) کار خود را انجام بدهد. این ایده که CPU سریع در حین انتظار به کارهای دیگری بپردازد بیشتر به یک افسانه شبیه است و مشکلات خاص خود را دارد. مضاف بر این وقتی دو CPU همه‌منظوره، با یکدیگر در ارتباط باشند شرایط رقابتی (Race) رخ می‌دهد و به پروتکلهای پیچیده‌ای برای هماهنگ کردن آنها با یکدیگر نیاز خواهد بود. عموماً بهترین راهکار، ساده‌تر کردن پروتکل و محول نمودن کار به CPU اصلی است.

حال ببینیم مسئله فیدبک [یعنی اعلام طول پنجره یا هر مورد مشابه توسط سمت مقابل] در پروتکلهای بسیار سریع چگونه حل و فصل می‌شود. به دلیل وجود حلقه با تأخیر (نسبتاً) بالا، حتی‌الامکان باید از ایجاد فیدبک اجتناب شود زیرا زمان زیادی طول می‌کشد تا سیگنال گیرنده به فرستنده برگردد. مثالی از فیدبک، اعمال نظارت و کنترل نرخ ارسال با استفاده از پروتکل پنجرهٔ لغزان می‌باشد. برای رهایی از تأخیر زیادی که در اعلام طول پنجرهٔ گیرنده به فرستنده همیشه وجود دارد، در شبکه‌های سریع بهتر است از یک پروتکل مبتنی بر نرخ پایه (و بدون فیدبک) بهره گرفته شود. در چنین پروتکلی فرستنده می‌تواند به هر اندازه که تمایل دارد داده بفرستد ولی حق ندارد با نرخی بیشتر از مقداری که قبلاً توافق کرده‌اند، ارسال نماید.

مثال دیگری از فیدبک، الگوریتم «شروع آهسته ژاکوبسن» است. این الگوریتم چندین آزمایش انجام می‌دهد تا مشخص کند که شبکه از عهدهٔ چه مقدار داده بر می‌آید. [بخش ۶-۵-۹] در شبکه‌های بسیار سریع، انجام چند آزمایش (و حتی یکی دو آزمایش) برای ارزیابی چگونگی پاسخ شبکه، بخش بسیار بزرگی از پهنای باند را تلف می‌کند. الگوی کارآمدتر آن است که در همان ابتدای برقراری اتصال، فرستنده، گیرنده و شبکه همگی منابع مورد نیاز را رزرو نمایند. رزرو کردن منابع از قبل، این حسن بزرگ را دارد که ساده‌تر می‌توان مقدار لرزش (Jitter) را

کاهش داد. کوتاه سخن آن که حرکت به سوی شبکه‌های بسیار سریع، طراحی پروتکلها را اجباراً به طرف اتصال‌گرا شدن یا چیزی شبیه به آن سوق می‌دهد. البته اگر در آینده، پهنای باند آنقدر فراوان و کم اهمیت شود که کسی نگران هدر رفتن مقدار قابل توجهی از آن نباشد، اصول طراحی نیز بسیار متفاوت خواهد بود.

ساختار و قالب هر بسته در شبکه‌های گیگابیتی حائز اهمیت است. در سرآیند هر بسته حتی‌الامکان باید تعداد بسیار کمی فیلد تعریف شده باشد تا زمان پردازش آن کاهش یابد. در ضمن این فیلدها باید بقدر کافی بزرگ باشند تا کار خود را بدون محدودیت و نیاز به پردازشهای اضافی انجام بدهند. همچنین هر فیلد باید به صورت کلمه (یعنی ۱ بایتی یا ۲ و ۴ و ۸ و ۱۶ بایتی) تعریف شوند تا پردازش آنها بسیار ساده باشد. منظور مان از فیلد «به قدر کافی بزرگ» آن است که مشکلاتی نظیر برگشت شمارهٔ ترتیب به مقادیر تکراری یا ناتوانی گیرنده از اعلام اندازهٔ بزرگ پنجرهٔ خود و مسائلی از این قبیل پدید نیاید.

بعلاوه بایستی برای بخش سرآیند و بخش داده‌ها در هر بسته ، دو کُد کشف خطای مجزا و مستقل محاسبه و درج شود، به دو دلیل: اول آن که در صورت عدم نیاز به نظارت بر خطای داده‌ها [مثلاً برای داده‌های صدا و تصویر]، بتوان فقط سرآیند بسته را از لحاظ صحت مقادیر بررسی کرد. دوم آن که بتوان قبل از کپی کردن داده‌ها در فضای پروسهٔ کاربر از صحت سرآیند مطمئن شد. مطلوب آن است که کنترل صحت بخش داده در خلال کپی کردن آنها در فضای پروسهٔ کاربر انجام شود ولیکن اگر سرآیند بسته صحیح نباشد ممکن است داده‌ها به اشتباه برای پروسهٔ دیگری کپی شوند. برای اجتناب از کپی اشتباهی داده‌ها، داشتن دو کد کشف خطای مستقل الزامی است. در این صورت می‌توان بررسی صحت بخش داده را به زمان کپی آن موکول کرد.

طول حداکثر داده‌ها در هر بسته باید بزرگ باشد تا حتی در مواجهه با تأخیر زیاد، کارآیی عملیات بالا باشد. هر چه طول داده‌ها بیشتر باشد درصدی از پهنای باند که صرف سرآیند هر بسته می‌شود کاهش خواهد یافت. بسته‌های ۱۵۰۰ بایتی برای شبکه‌های گیگابیتی بسیار کوچکند.

ویژگی ارزشمند دیگر آن است که فرستنده بتواند به همراه تقاضای برقراری اتصال، مقداری داده نیز بفرستد. بدین ترتیب در زمان رفت و برگشت صرفه‌جویی می‌شود.

در آخر، چند کلمه صحبت در خصوص نرم‌افزار پروتکل خالی از لطف نیست. اندیشهٔ مثبت بر موفقیت‌آمیز بودن عمل متمرکز می‌شود در حالی که بسیاری از پروتکلهای قدیمی بیشتر بر این محور تکیه دارند که اگر خطایی رخ داد (مثلاً بسته‌ای در بین راه از بین رفت) چه باید کرد. برای آن که اجرای پروتکلها سریع شود، طراح باید هدف خود را بر آن بگذارد که وقتی همه چیز به خوبی پیش می‌رود به حداقل پردازش نیاز باشد. سپس بدان بپردازد که چگونه می‌توان در هنگام بروز خطا حجم پردازشها را به حداقل رساند. [اولویت با حداقل بودن پردازش در شرایط عادی است چرا که شرایط غیرعادی به ندرت پیش می‌آید. -م]

مسئلهٔ دیگر در طراحی نرم‌افزار پروتکلها به حداقل رساندن زمان کپی‌برداری [انتقال داده‌ها در حافظه] است. همانگونه که قبلاً دیدیم، کپی‌برداری از داده‌ها عامل عمدهٔ تحمیل سربار است. آرمانی آنست که سخت‌افزار، بستهٔ ورودی را در قالب یک بلوک یکپارچه و به صورت یکجا به حافظه منتقل نماید. سپس نرم‌افزار باید این بسته را تنها با یک عمل کپی بلوکی به بافر کاربر منتقل کند. براساس چگونگی عملکرد حافظهٔ نهان CPU ممکن است اجتناب از حلقهٔ تکرار در برنامه‌نویسی به منظور کپی داده‌ها، مطلوبتر باشد. به عبارت دیگر برای کپی کردن ۱۰۲۴ بایت داده، بهتر آن است که به جای حلقه از ۱۰۲۴ دستورالعمل MOVE پشت سرهم (یا ۱۰۲۴ جفت دستورالعمل LOAD و STORE) استفاده شود! به لحاظ اهمیت حیاتی، زیربرنامهٔ کپی بایستی با کدهای اسمبلی نوشته شود مگر آن که بتوان به طریقی کامپایلر را به منظور تولید کد بهینه و سریع تنظیم کرد.

۶-۷ خلاصه

لایهٔ انتقال کلید آشنایی با پروتکلهای لایه‌ای است. این لایه خدمات متنوعی را عرضه می‌کند ولی مهمترین آنها ایجاد یک استریم مطمئن، انتها به انتها و اتصال‌گرا بین گیرنده و فرستنده به منظور انتقال دنباله‌ای از بایتهاست. دسترسی به خدمات این لایه از طریق یکسری توابع اولیه و پایه صورت می‌گیرد که این توابع برقراری اتصال، استفاده از آن (جهت ارسال و دریافت) و خاتمهٔ اتصال را امکان‌پذیر کرده‌اند. یکی از واسطهای رایج برای لایهٔ انتقال با نام «سوکتهای برکلی» (Berkeley Socket) ارائه شده است.

پروتکلهای لایهٔ انتقال باید بتوانند بر اتصالاتی که در یک شبکهٔ نامطمئن ایجاد می‌شوند مدیریت کنند. برقراری اتصال از آن جهت پیچیده است که امکان دارد بسته‌های تکراری و معلق در زیرشبکه، در لحظه‌ای به گیرنده برسند که او را به اشتباه بیندازند. برای حل و فصل این مشکل، باید برای برقراری یک اتصال از روش دست‌تکانی سه‌مرحله‌ای (3-Way Handshake) استفاده شود. ختم یک اتصال ساده‌تر از برقراری آن است ولیکن به دلیل «مشکل دو سپاه» (Two-Army Problem) چندان هم پیش پا افتاده نیست.

حتی وقتی که لایهٔ شبکه کاملاً مطمئن و بدون خطاست باز هم لایهٔ انتقال کار زیادی باید انجام بدهد. این لایه باید وظیفهٔ ارائه خدمات اولیه (توابع اولیه)، مدیریت اتصالات و تایمرها را بر عهده بگیرد و «اعتبار» تخصیص بدهد.

اینترنت در لایهٔ انتقال دو پروتکل دارد: UDP و TCP. UDP پروتکلی بدون‌اتصال است که در حقیقت همان خصوصیات پروتکل IP را دارد، با این ویژگی اضافی که بتوان بسته‌های IP را که همگی دارای آدرس مشترک هستند، بین پروسه‌های یک ماشین مالتی‌پلکس و دی‌مالتی‌پلکس کرد. از UDP می‌توان برای تعامل بین سرویس‌دهنده و مشتری بهره گرفت (مثلاً به کمک RPC). همچنین می‌توان از UDP برای ساختن پروتکلهای بی‌درنگ همانند RTP استفاده کرد.

پروتکل اصلی لایهٔ انتقال در اینترنت، TCP است. این پروتکل یک استریم از بایتها را (به صورت دو طرفه) بین دو پروسه ایجاد کرده و در اختیار می‌گذارد. به یک واحد اطلاعاتی که توسط TCP تولید می‌شود اصطلاحاً «قطعه» (Segment) گفته می‌شود. هر قطعه دارای یک سرآیند ۲۰ بایتی است. قطعات ارسالی ممکن است توسط مسیریابهای اینترنت قطعه قطعه شوند فلذا ماشین میزبان باید آمادگی بازسازی آنها را داشته باشد. کارهای بسیار زیادی نیز برای بهینه سازی کارآیی TCP صورت گرفته و بدین منظور از الگوریتمهایی نظیر «ناگل»، «کلارک»، «ژاکوبسن»، «کارن» استفاده شده است. لینکهای بی‌سیم پیچیدگیهای متعددی را به TCP تحمیل می‌کنند. «TCP تراکنشی» (Transaction TCP) گونهٔ توسعه یافتهٔ TCP است که تعامل سریع بین سرویس‌دهنده و مشتری و کاهش تعداد بسته‌ها را بر عهده دارد.

کارآیی شبکه عموماً متأثر از سربار پروتکل و پردازش TPDU است و در سرعتهای بالا این وضعیت به مراتب بدتر می‌شود. پروتکلها بایستی به نحوی طراحی شوند که تعداد TPDUها، دفعات تعویض متن (Context Switch) و دفعات کپی‌برداری از TPDU را به حداقل برساند. برای شبکه‌های گیگابیتی، پروتکلهای ساده مورد توجه هستند.

مسائل

۱. در مثالی که در شکل ۶-۲ در خصوص عملکردهای پایهٔ انتقال (Primitives) ارائه کردیم، عمل LISTEN یک نوع فراخوانی متوقف‌کننده (Blocking) است. آیا این رفتار واقعاً لازم بوده است؛ اگر نه، شرح بدهید که چگونه می‌توان از یک تابع غیرمتوقف‌کننده (nonblocking) به جای آن استفاده کرد و این روش چه مزیتی

بر الگویی که در متن تشریح شد ، دارد؟

۲. در مدل شکل ۶-۴ فرض بر آنست که احتمالاً برخی از بسته‌ها در لایهٔ شبکه از بین می‌روند و هر بسته باید بطور مستقل اعلام وصول شود. فرض کنید که لایهٔ شبکه صد در صد مطمئن و بدون خطاست و هیچ بسته‌ای از دست نمی‌رود. در این حالت چه تغییری در شکل ۶-۴ لازم است؟

۳. در هر دو بخش از کُد برنامهٔ شکل ۶-۶ بدین نکته اشاره شده که مقدار SERVER_PORT باید در سرویس‌دهنده و مشتری یکسان باشد. چرا این موضوع اینقدر اهمیت دارد؟

٤. فرض کنید که برای تولید مقدار اولیه برای شمارهٔ ترتیب از روش مبتنی بر ساعت استفاده شده و شمارندهٔ ساعت ۱۵ بیتی است. ساعت در هر ۱۰۰ میلی‌ثانیه یکبار تیک می‌زند و حداکثر طول عمر بسته‌ها ۶۰ ثانیه است. در چه مواقعی به هماهنگ‌سازی دوباره (Resynchronization) نیاز است: (پاسخ خود را برای موارد ذیل محاسبه کنید)

الف) در بدترین حالت

ب) وقتی که برای ارسال داده‌ها در هر دقیقه ۲۴۰ شمارهٔ ترتیب مصرف می‌شود.

۵. چه لزومی دارد که حداکثر طول عمر بسته‌ها یعنی T، باید آنقدر طولانی باشد تا مطمئن شویم که نه تنها بسته بلکه حتی پیغامهای اعلام وصول آنها (ACK) نیز از بین رفته‌اند؟

۶. تصور کنید که برای برقراری اتصال به جای روش «دست‌تکانی سه‌مرحله‌ای» از «دست‌تکانی دومرحله‌ای» استفاده شده باشد. (به عبارت دیگر به پیام سوم نیازی نباشد.) آیا امکان بروز بن‌بست وجود دارد. یا نشان بدهید که بن‌بستی بوجود نمی‌آید یا مثالی از بن‌بست ارائه بدهید.

۷. مسئله «دو سپاه» را به صورت عمومی «**n سپاه**» (n-Army) در نظر بگیرید که فقط توافق دو سپاه آبی برای حمله، پیروزی آنها را تضمین می‌کند. آیا پروتکلی وجود دارد که براساس آن سپاه آبی به یقین پیروز میدان شود؟

۸. مشکل جبران از کارافتادگی یک ماشین میزبان و بازگرداندن آن به حالت طبیعی را مدّ نظر قرار بدهید؛ (شکل ۶-۱۸). هرگاه بتوان فاصلهٔ زمانی بین نوشتن داده‌ها در پروسه و ارسال پیغام اعلام وصول (ACK) یا برعکس را کم کرد، دو تا از بهترین استراتژیهای فرستنده و گیرنده که احتمال شکست پروتکل را به حداقل می‌رساند، کدامند؟

۹. آیا در واحد انتقال معرفی شده در شکل ۶-۲۰ امکان بروز بن‌بست وجود دارد؟

۱۰. شخصی که واحد انتقال شکل ۶-۲۰ را پیاده‌سازی کرده است تصمیم گرفته که در پروسیجر ***sleep*** یک شمارنده بگذارد تا در خصوص آرایهٔ conn آمارگیری کند. از جملهٔ این آمارها، تعداد اتصالهایی است که در هر یک از هفت حالت ممکن قرار دارد. (n_i, $i=1,2,...,7$) پس از نوشتن برنامه‌ای مفصل به زبان فرترن برای تحلیل داده‌های جمع‌آوری شده، برنامه‌نویس ما مشاهده می‌کند که رابطهٔ $\Sigma n_i = MAX_CONN$ همیشه صادق است. آیا روابط این چنین دیگری (که حاصل ثابتی داشته باشند) براساس این هفت «متغیر حالت» وجود دارد؟ [حالات هفتگانهٔ n_i را در شکل ۶-۲۱ مرور کنید. -م]

۱۱. اگر یک کاربر با استفاده از واحد انتقال نشان داده شده در شکل ۶-۲۰ یک پیام با طول صفر بفرستد چه اتفاقی می‌افتد؟

۱۲. تمام رخدادهایی را که ممکن است در واحد انتقال شکل ۶-۲۰ اتفاق بیفتد، در نظر بگیرید. حال بگویید که آیا این رخدادها هنگامی که کاربر در حالت *sending* متوقف مانده است باز هم معتبرند؟

۱۳. مزایا و معایب استفاده از روش مبتنی بر اعتبار را در مقایسه با پروتکلهای پنجرهٔ لغزان تشریح کنید.

۱٤. چرا به وجود UDP نیاز است؟ آیا اگر پروسه‌های کاربری داده‌های خود را در درون بسته‌های IP جاسازی و ارسال کند کافی نیست؟

۱۵. یک پروتکل بسیار سادهٔ لایهٔ کاربرد را در نظر بگیرید که بر روی UDP کار می‌کند و امکان آنرا فراهم آورده تا مشتری بتواند یک فایل را از سرویس‌دهندهٔ راه دور دریافت نماید. آدرس سرویس‌دهنده (یعنی آدرس IP و پورت) کاملاً مشخص است. مشتری ابتدا یک تقاضا حاوی نام فایل درخواستی برای سرویس‌دهنده می‌فرستد. سرویس‌دهنده در پاسخ دنباله‌ای از بسته‌های داده را که هر یک بخشی از فایل درخواستی را حمل می‌کنند، برای مشتری ارسال می‌دارد. برای اطمینان از صحّت داده‌ها و همچنین حفظ ترتیب بسته‌های دریافتی، سرویس‌دهنده و مشتری از پروتکل «توقف و انتظار» (Stop & Wait) بهره گرفته‌اند. صرفنظر از مسئله کارآیی، آیا مشکل دیگری در این پروتکل می‌بینید؟ به دقت در مورد امکان از کار افتادن (crashing) پروسه‌ها فکر کنید.

۱۶. یک برنامهٔ مشتری، از طریق یک فیبرنوری به طول ۱۰۰ کیلومتر و سرعت یک گیگابیت بر ثانیه، تقاضایی ۱۲۸ بایتی (RPC) برای سرویس‌دهنده می‌فرستد. کارآیی خط در خلال این فراخوانی از راه دور چقدر است؟

۱۷. وضعیت توصیف شده در مسئله قبلی را مدّ نظر قرار بدهید. حداقل زمان پاسخ سرویس‌دهنده را برای خطوط 1-Gbps و 1-Mbps محاسبه نمائید. چه نتیجه‌ای می‌توانید بگیرید؟

۱۸. UDP و TCP هر دو برای مشخص کردن پروسهٔ تحویل گیرنده پیام از شمارهٔ پورت استفاده می‌کنند. دو دلیل بیاورید که چرا این دو پروتکل برای مشخص کردن پروسه‌ها، شناسه‌های جدیدی تعریف کرده‌اند (یعنی شمارهٔ پورتها) و از شناسهٔ پروسه (Process ID) که از قبل [در هستهٔ سیستم عامل] وجود دارد استفاده نمی‌کنند؟

۱۹. حداقل اندازهٔ کل یک بستهٔ TCP (شامل سربار IP و TCP، بدون در نظر گرفتن سربار لایهٔ پیوند داده) چقدر است؟

۲۰. فرآیند قطعه‌قطعه کردن دیتاگرام و بازسازی آنها بر عهدهٔ IP است و از دید TCP مخفی می‌ماند. آیا این قضیه بدین معنا تلقی می‌شود که TCP نباید نگران تحویل نامرتب داده‌ها باشد؟

۲۱. برای انتقال صدا با کیفیت CD، از پروتکل RTP استفاده می‌شود و باید در هر ثانیه ۴۴۱۰۰ جفت نمونهٔ ۱۶ بیتی ارسال شود. (هر یک از این جفت نمونه‌ها برای یکی از کانالهای استریو است.) RTP در هر ثانیه باید چند بسته ارسال کند؟

۲۲. آیا می‌توان کُد اجرایی RTP را در کنار UDP درون هستهٔ سیستم عامل قرار دارد؟ پاسخ خود را شرح بدهید.

۲۳. به یک پروسه بر روی ماشین میزبان ۱ شمارهٔ پورت p انتساب داده شده است. همچنین به پروسهٔ دیگری بر روی ماشین ۲، شمارهٔ پورت q منتسب شده است. آیا این امکان وجود دارد که در یک زمان دو یا چند اتصال TCP بین این دو پورت برقرار شود؟

۲٤. در شکل ۶-۲۹ می‌بینیم که به غیر از فیلد ۳۲ بیتی Acknowledgement، در چهارمین کلمه یک بیت دیگر با نام ACK وجود دارد. آیا این بیت واقعاً کاری انجام می‌دهد؟ چرا بله و چرا نه؟

۲۵. حداکثر طول فیلد حمل داده (Payload) در یک قطعهٔ TCP، ۶۵۴۹۵ بایت است. این عدد عجیب از کجا آمده است؟

۲۶. در شکل ۶-۳۳، دو راه وارد شدن به حالت SYN RCVD را تشریح نمایید.

۲۷. اشکال بالقوهٔ «الگوریتم ناگل» (Nagle) را در بکارگیری آن بر روی شبکه‌ای که شدیداً با مشکل ازدحام مواجه شده، تشریح کنید.

۲۸. تأثیر استفاده از «الگوریتم شروع آهسته» (Slow start algorithm) را بر روی خطی بدون ازدحام و با تأخیر رفت و برگشت ده میلی‌ثانیه، مد نظر قرار بدهید. پنجرهٔ دریافت 24KB و حداکثر طول هر قطعه 2KB است. چقدر طول می‌کشد تا به اندازهٔ یک پنجرهٔ کامل (یعنی ۲۴KB) ، داده ارسال شود؟

۲۹. فرض کنید که «پنجرهٔ ازدحام TCP» به 18KB تنظیم شده باشد و انقضای مهلت تایمر رخ بدهد. اگر چهار ارسال بعدی موفق باشد اندازهٔ پنجره چقدر است؟ (فرض کنید که حداکثر طول قطعه 1KB می‌باشد.)

۳۰. فرض نمایید زمان رفت و برگشت (یعنی RTT) ، فعلاً ۳۰ میلی‌ثانیه است و سه پیغام ACK بعدی به ترتیب پس از ۲۶ ، ۳۲ و ۲۴ میلی‌ثانیه دریافت شوند. با فرض $\alpha = 0.9$ در الگوریتم ژاکوبسن، مقدار جدید RTT چقدر است؟

۳۱. یک ماشین مبتنی بر TCP ، به اندازهٔ یک پنجرهٔ ۶۵۵۳۵ بایتی بر روی یک کانال 1-Gbps ، داده می‌فرستد. کانال فقط در یک جهت ده میلی‌ثانیه تأخیر دارد. حداکثر توان خروجی (Throughput) قابل حصول چقدر است؟ کارآیی خط چقدر است؟

۳۲. فرض کنید حداکثر طول عمر بسته‌ها ۱۲۰ ثانیه است. یک ماشین میزبان حداکثر با چه سرعتی می‌تواند بسته‌های TCP با ۱۵۰۰ بایت داده را بفرستد در حالی که خطر تکرار شمارهٔ ترتیب تهدیدکننده نباشد؟ سربار ناشی از سرآیند TCP ، IP و اترنت را هم در نظر بگیرید. فرض کنید که فریمهای اترنت را می‌توان پیاپی و بی‌وقفه فرستاد.

۳۳. در یک شبکه اندازهٔ حداکثر هر TPDU ، ۱۲۸ بایت و حداکثر طول عمر آن ۳۰ ثانیه است. اگر شمارهٔ ترتیب هر TPDU هشت بیتی باشد، حداکثر نرخ ارسال در هر اتصال را محاسبه نمایید.

۳٤. فرض کنید زمان دریافت یک TPDU را اندازه‌گیری می‌کنید. وقتی یک وقفه رخ می‌دهد (وقفهٔ دریافت بسته) شما بلافاصله ساعت سیستم را برحسب میلی‌ثانیه می‌خوانید. وقتی TPDU کاملاً پردازش شد مجدداً اقدام به خواندن ساعت می‌کنید. این کار را یک میلیون بار تکرار کرده‌اید و ۲۷۰۰۰۰ بار زمان را صفر و ۷۳۰۰۰۰ بار زمان را ۱ میلی‌ثانیه بدست آورده‌اید. با این اندازه‌گیریها زمان دریافت یک TPDU چقدر است؟

۳۵. یک CPU دستورالعملهای ماشین را با سرعت 1000-MIPS اجرا می‌کند. داده‌ها را می‌توان به صورت کلمات ۶۴ بیتی کپی کرد و کپی هر کلمه معادل زمان اجرای ۱۰ دستورالعمل طول می‌کشد. اگر هر بستهٔ ورودی نیاز به چهار بار کپی‌برداری داشته باشد، آیا این سیستم قادر است از عهدهٔ یک خط 1-Gbps برآید؟ برای سادگی فرض کنید تمام دستورالعملهای ماشین حتی آنهایی که از حافظه می‌خوانند یا در آن می‌نویسند، همگی با سرعت 1000-MIPS اجرا شوند.

۳۶. برای رهایی از مشکل تکراری شدن شمارهٔ ترتیب بسته‌ها (در حالی که ممکن است بسته‌های قدیمی هنوز در زیرشبکه سرگردان باشند) می‌توان از شماره‌های ۶۴ بیتی استفاده کرد. با این وجود از دیدگاه تئوری می‌توان بکمک فیبرنوری به نرخ 75Tbps (75000 Gbps) رسید. حداکثر طول عمر بسته‌ها چقدر باشد که در شبکه‌های آینده با سرعت 75Tbps شماره‌های ترتیب ۶۴ بیتی تکرار نشوند؟ (فرض کنید که همانند TCP ، هر بایت دارای یک شمارهٔ ترتیب است.)

۳۷. یکی از مزیتهای استفاده از RPC بر روی UDP را نسبت به استفاده از «TCP تراکنشی» بیان کنید. یک مزیت T/TCP را نسبت به RPC عنوان نمایید.

۳۸. در شکل ۶-۴۰-الف مشاهده می‌کنید که برای تکمیل یک فراخوانی راه دور (RPC) به مبادله حداقل ۹ بسته نیاز است. آیا وضعیت دیگری وجود دارد که دقیقاً به ۱۰ بسته نیاز باشد؟

۳۹. در بخش ۶-۶-۵ محاسبه کردیم که خطی یک گیگابیتی ۸۰۰۰۰ بسته در هر ثانیه به درون ماشین میزبان منتقل می‌کند. همچنین فرض کردیم که برای پردازش هر بسته فقط ۶۲۵۰ دستورالعمل اجرا گردد و نیم دیگر از توان پردازشی CPU برای کاربردهای دیگر کنار گذاشته شود. این محاسبات بر مبنای بسته‌های ۱۵۰۰ بایتی بود. همین محاسبات را برای بسته‌های ۱۲۸ بایتی (هم اندازهٔ بسته‌های ARPANET) انجام بدهید. در هر دو حالت فرض را بر آن بگذارید که سربار کل در اندازهٔ بسته لحاظ شده است. [یعنی نیازی به اضافه کردن طول سرآیندها به مقادیر ۱۲۸ یا ۱۵۰۰ نیست.]

۴۰. برای شبکه‌ای 1-Gbps که طول کانال آن ۴۰۰۰ کیلومتر است، عامل اصلی محدودیت تأخیر خط است نه پهنای باند. حال یک MAN را در نظر بگیرید که میانگین فاصلهٔ مبداء و مقصد، ۲۰ کیلومتر است. در چه نرخ ارسالی تأخیر رفت و برگشت (ناشی از سرعت نور) معادل با تأخیر انتقال یک بستهٔ یک کیلوبایتی است؟ [تاخیر انتشار ناشی از سرعت نور است درحالیکه تاخیر انتقال متاثر از نرخ ارسال]

۴۱. حاصلضرب پهنای باند در تأخیر را برای شبکه‌های زیر محاسبه کنید: (۱) T1 (با نرخ 1.5Mbps) (۲) اترنت (10Mbps) (۳) T3 (45Mbps) (۴) STS-3 (155 Mbps). زمان رفت و برگشت یعنی RTT را ۱۰۰ میلی‌ثانیه فرض کنید. به خاطر داشته باشید که در سرآیند TCP فقط ۱۶ بیت جهت اعلام طول پنجره رزرو شده است. از محاسبات خود به چه نتیجه‌ای می‌رسید؟

۴۲. حاصلضرب پهنای باند در تأخیر یک کانال ماهواره‌ای همگام با زمین (ماهوارهٔ نوع GEO) چقدر است؟ اگر تمام بسته‌ها با احتساب سربار، ۱۵۰۰ بایتی باشند اندازهٔ فیلد پنجره در هر بسته باید چقدر باشد؟

۴۳. سرویس‌دهندهٔ فایل نشان داده شده در شکل ۶-۶ بسیار ضعیف عمل می‌کند و می‌توان بهبودهایی را در آن ایجاد کرد. اصلاحات زیر را در آن اعمال نمایید:

الف) به برنامهٔ مشتری آرگومان سومی اضافه کنید که محدودهٔ بایتهائی که باید ارسال شوند را مشخص کند.

ب) به برنامهٔ مشتری آرگومان پرچم w- را اضافه کنید تا بتوان فایلی را بر روی سرویس‌دهنده نوشت.

۴۴. برنامهٔ شکل ۶-۲۰ را به نحوی اصلاح کنید که عملیات جبران خطا (Error Recovery) را نیز انجام بدهد. یک بستهٔ نوع جدید به نام reset تعریف کنید که فقط زمانی دریافت می‌شود که طرفین اقدام به برقراری اتصال کرده باشند ولی به هر دلیلی هیچیک از طرفین آن را قطع نکرده باشند. این رخداد که بطور همزمان در طرفین یک اتصال بروز می‌کند بدین معناست تمام بسته‌هایی که قبلاً ارسال شده‌اند یا دریافت گردیده‌اند و یا از بین رفته‌اند ولیکن به هر حال در زیرشبکه نیستند.

۴۵. برنامه‌ای بنویسید که به جای سیستم مبتنی بر اعتبار (credit) که در واحد انتقال شکل ۶-۲۰ از آن استفاده شد، مدیریت بافرها را مبتنی بر پروتکل پنجرهٔ لغزان (برای کنترل جریان) شبیه سازی کند. باید پروسه‌های لایهٔ بالاتر بتوانند اتصالاتی را باز کنند، داده بفرستند و نهایتاً اتصال را ببندند. برای ساده شدن برنامه فرض کنید که کل داده‌ها در یک جهت و از ماشین A به ماشین B جریان دارند. در برنامهٔ خود استراتژیهای مختلف تخصیص بافر را در ماشین B بیازمائید، مثلاً روش تخصیص پیشاپیش بافر برای هر اتصال را با روش

معمولی بافرسازی (یعنی استفاده از یک بافر بزرگ) مقایسه کرده و توان خروجی آن دو مقایسه کنید.

۴۶. یک سیستم گپ زنی (chat) طراحی کنید که امکان گفتگو و محاورهٔ چندین گروه از کاربران را فراهم کند. این سیستم از سه بخش تشکیل شده است: (۱) هماهنگ‌کنندهٔ گفتگو (Chat Coordinator) (۲) سرویس‌دهندهٔ گفتگو (Chat Server) (۳) برنامهٔ مشتری (Chat Client). سیستم هماهنگ‌کنندهٔ گفتگو در آدرس شناخته‌شده‌ای از شبکه قرار گرفته است. سیستم هماهنگ کننده با برنامه‌های مشتری به روش UDP تبادل داده می‌کند و برای هر نشستی که ایجاد می‌شود یک سرویس‌دهندهٔ گفتگو (Chat Server) تنظیم می‌کند. یعنی به ازای هر نشست یک سرویس‌دهندهٔ گفتگو وجود دارد. [یک نشست را مجموعه‌ای از کاربران در نظر بگیرید که حول یک موضوع خاص گفتگو می‌کنند.] از دیگر وظائف سیستم هماهنگ‌کننده، نگهداری یک دایرکتوری از نشستهای موجود است. سرویس‌دهندهٔ گفتگو برای ارتباط با هر مشتری آن نشست، از TCP بهره می‌گیرد. برنامهٔ مشتری نیز به کاربران اجازه می‌دهد که یک نشست ایجاد کنند، به یک نشست موجود بپیوندند یا یک نشست را ترک کنند. کُد برنامهٔ سیستم هماهنگ‌کننده، سرویس‌دهنده و مشتری را طراحی و پیاده‌سازی نمائید.



لایه کاربرد

بعد از به سرانجام رساندن همهٔ مقدمات، اکنون به لایه‌ای رسیده‌ایم که تمام کاربردهای شبکه در آن قرار دارد: لایهٔ کاربرد (application layer). لایه‌های زیرین لایهٔ کاربرد فقط برای سرویس دادن به این لایه هستند، و هیچ کار واقعی برای کاربران انجام نمی‌دهند. در این فصل با کاربردهای واقعی شبکه آشنا خواهید شد.

با این حال در لایهٔ کاربرد هم به پروتکلهای پشتیبانی‌کننده، پروتکلهایی که بار وظایف را بر گردن می‌گیرند، نیاز داریم. بهمین دلیل، برای شروع یکی از این پروتکلها را بررسی خواهیم کرد. این پروتکل که نام آن DNS است، نامگذاری در اینترنت را بر عهده دارد. پس از آن، سه تا از کاربردهای واقعی شبکه را مورد بررسی قرار خواهیم داد: پُست الکترونیک (ایمیل)، تارنمای جهانی (یا باختصار، وب)، و چندرسانه‌ای.

۷-۱ سیستم نام ناحیه - DNS

با اینکه از نظر تئوری برنامه‌ها می‌توانند برای تماس با کامپیوترها، صندوق‌های پستی و منابع دیگر از آدرس شبکهٔ (مثلاً، IP) آنها استفاده کنند، حفظ کردن این قبیل آدرسها برای افراد دشوار است. همچنین اگر آدرس ایمیل «حسن» hasan@128.111.24.41 باشد، و ISP یا سازمان متبوع وی تصمیم بگیرد کامپیوتر سرویس‌دهنده پست الکترونیک خود را به آدرس IP دیگری منتقل کند، آدرس ایمیل «حسن» نیز عوض خواهد شد. بهمین دلیل برای تفکیک نام ماشینها از آدرس آنها، استفاده از نامهای معمولی باب شد. به این ترتیب، آدرس ای بل «حسن» چیزی شبیه hasan@art.ucsb.edu خواهد شد. با این وجود، کامپیوترها فقط آدرسهای عددی را می‌فهمند، پس باید مکانیزمی برای تبدیل اسامی معمولی به آدرسهای شبکه فراهم کنیم. در این قسمت روش تبدیل نامهای معمولی به آدرس عددی را در اینترنت بررسی خواهیم کرد.

سالها قبل در آرپانت فایلی وجود داشت بنام hosts.txt ، که نام کامپیوترها و آدرس IP آنها در این فایل لیست می‌شد. کامپیوترهای شبکه هر شب این فایل را از جایی که قرار داشت، می‌خواندند و خود را به روز می‌کردند. برای شبکه‌ای با دهها (و یا صدها) کامپیوتر این روش بخوبی کار می‌کرد.

ولی وقتی تعداد کامپیوترهای شبکه از مرز هزاران PC و کامپیوتر بزرگ گذشت، همه دریافتند که این روش دیگر جوابگو نیست. اولین دلیل آن بود که اندازهٔ چنین فایلی بشدت بزرگ می‌شد، ولی از آن مهمتر مشکل نامهای تکراری بود که ضرورت یک مدیریت مرکزی را اجتناب‌ناپذیر می‌کرد (چیزی که بزرگی و بار شبکه آنرا ناممکن می‌کرد). برای غلبه بر این مشکلات بود که DNS (**سیستم‌نام‌ناحیه** - Domain Name System) اختراع شد.

ایدهٔ اصلی DNS یک روش نامگذاری سلسله مراتبی بر اساس ناحیه‌ها بود، که بصورت یک پایگاه اطلاعاتی

توزیع‌یافته پیاده‌سازی می‌شد. هدف اولیهٔ این سیستم تبدیل نام کامپیوترها و آدرسهای ایمیل (پست الکترونیک) به آدرسهای IP بود، ولی می‌توانست کاربردهای دیگری هم داشته باشد. DNS در RFC 1034 و RFC 1035 تعریف شده است.

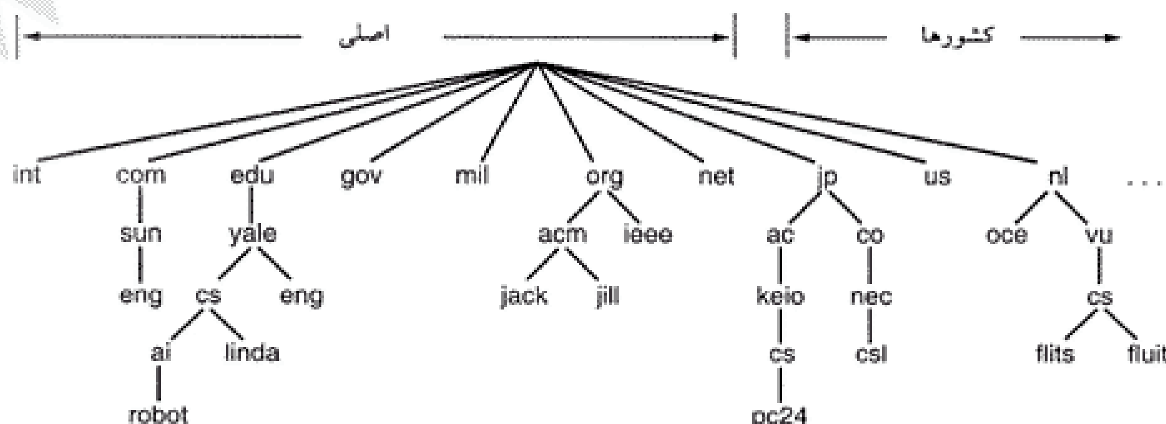
روش کار DNS خیلی خلاصه چنین است: برای تبدیل یک نام به آدرس IP ، برنامه یک تابع کتابخانه‌ای بنام **تبدیل‌کننده** (resolver) را فراخوانی می‌کند، و نام موردنظر را بصورت پارامتر به آن می‌دهد. تابع *gethostbyname* در شکل ۶-۶ نمونه‌ای از یک تبدیل‌کننده است. تبدیل‌کننده یک بستهٔ UDP به سرویس‌دهنده DNS محلی می‌فرستد، که این DNS آدرس IP معادل نام خواسته شده را یافته و به تبدیل‌کننده برمی‌گرداند، که آن هم بنوبهٔ خود آدرس را به برنامهٔ فراخوانی‌کننده تحویل می‌دهد. برنامه هم پس از بدست آوردن آدرس IP کامپیوتر مقصد، می‌تواند با آن ارتباط TCP برقرار کرده یا بسته‌های UDP به آن بفرستد.

۷-۱-۱ فضای نام DNS

مدیریت مجموعه‌ای بزرگ و دائماً در تغییر از نامها بهیچوجه کار ساده‌ای نیست. در سیستم پُست، مدیریت نامها از طریق اجبار افراد به نوشتن نام کشور، استان (یا ایالت)، شهر، خیابان و شمارهٔ پلاک مقصد انجام می‌شود. با این روش دیگر «پلاک ۷، خیابان آزادی، تهران» هرگز با «پلاک ۷، خیابان آزادی، اصفهان» اشتباه نخواهد شد. DNS هم به همین روش کار می‌کند.

اینترنت به بیش از ۲۰۰ **ناحیهٔ سطح بالا** (top-level domain) (که هر کدام تعداد زیادی کامپیوتر را در برمی‌گیرند) تقسیم شده است. هر ناحیه به چندین **زیرناحیه** (subdomain) ، و آنها نیز بنوبهٔ خود به زیرناحیه‌های کوچکتر، تقسیم می‌شوند. این سلسله مراتب را می‌توان بصورت یک درخت نمایش داد (شکل ۷-۱ را ببینید). ناحیه‌هایی که زیرناحیه ندارند، برگهای این درخت را تشکیل می‌دهند. هر یک از این برگ‌ها می‌تواند یک کامپیوتر، یک شبکهٔ کوچک (با چند کامپیوتر)، و یا شرکتی بزرگ (با هزاران کامپیوتر) باشد.

ناحیه‌های سطح بالا بر دو گونه‌اند: عمومی و کشورها. ناحیه‌های عمومی اولیه عبارت بودند از: *com* (مخفف commercial ، تجاری)، *edu* (مخفف educational ، مؤسسات آموزشی)، *gov* (مخفف goverment ، ادارات دولتی)، *int* (مخفف international ، مؤسسات بین‌المللی)، *mil* (مخفف military ، نظامی)، *net* (مخفف network provider ، شرکتهای خدمات شبکه و اینترنت)، و *org* (مخفف organization ، مؤسسات غیرانتفاعی). هر کشور نیز دارای یک ناحیهٔ خاص (در ناحیهٔ کشورها) است، که در استاندارد ISO 3166 تعریف شده است.



شکل ۷-۱. بخشی از فضای نام ناحیه در اینترنت.

در نوامبر ۲۰۰۰، ICANN چهار ناحیهٔ سطح بالای جدید را برای مصارف عمومی تصویب کرد: *biz* (مخفف businesses ، مشاغل)، *info* (مخفف information ، شرکتهای اطلاعاتی)، *name* (نام افراد)، و *pro* (مخفف proffession ، صاحبان حِرَف مانند وکلا و پزشکان). علاوه بر آن، سه ناحیهٔ سطح بالای تخصصی نیز معرفی شد، که برخی از صنایع خاص می‌توانند از آنها استفاده کنند؛ این سه ناحیه عبارتند از: *aero* (مخفف aerospace ، صنایع هوا-فضا)، *coop* (مخفف co-operatives ، تعاونی‌ها)، و *museum* (موزه‌ها). به احتمال زیاد در آینده ناحیه‌های سطح بالای دیگری نیز اضافه خواهند شد.

از طرف دیگر هر چه اینترنت بیشتر تجاری می‌شود، بحث و جدل هم بالاتر می‌گیرد. مثلاً همین ناحیهٔ *pro* را در نظر بگیرید. این ناحیه برای صاحبان حِرَف که صلاحیت آنها تأئید شده باشد، در نظر گرفته شده است. اما حرفه‌ای کیست؟ و چه کسی باید صلاحیت‌ها را تأئید کند؟ یک پزشک یا وکیل مسلماً حرفه‌ایست، اما یک عکاس، معلم پیانو، شعبده‌باز، لوله‌کش، آرایشگر، خالکوب، آدمکش حرفه‌ای و یا فاحشه چطور؟ آیا اینها هم حرفه‌اند، و شایستهٔ دریافت ناحیهٔ *pro* ؟ و اگر پاسخ مثبت است، چه کسی صلاحیت داوطلبان را تأئید می‌کند؟

در کل، گرفتن ناحیهٔ سطح-دوم در یک ناحیهٔ سطح بالا ، مانند *name-of-company*.com ، ساده است: تنها کاری که باید کرد مراجعه به پایگاه اطلاعاتی ناحیهٔ سطح بالا (در اینجا *com*) و اطمینان از آزاد بودن نام موردنظر است. اگر مشکلی وجود نداشته باشد، درخواست‌کننده پول کمی بابت هزینهٔ سالیانه پرداخته، و آن نام را بدست می‌آورد. می‌توان به جرأت گفت که، اکنون تمام نامهای بامعنای انگلیسی در ناحیهٔ *com* گرفته شده‌اند (اگر باور ندارید، امتحان کنید!).

نام هر ناحیه بصورت مسیری رو به بالا و به سمت یک ریشه (که نامی ندارد) مشخص می‌شود. اجزای این نام با **نقطه** (که «دات» تلفظ می‌شود) از هم جدا می‌شوند. برای مثال، نام ناحیهٔ بخش مهندسی شرکت سان میکروسیستمز می‌تواند *eng.sun.com.* باشد (در حالیکه همین نام در سیستم عامل یونیکس بصورت */com/sun/eng* نوشته می‌شود). دقت کنید که با این روش نامگذاری دیگر نام ناحیهٔ بخش مهندسی سان میکروسیستمز با دانشکدهٔ مهندسی دانشگاه ییل (که مثلاً *eng.yale.edu.* است) اشتباه نخواهد شد.

نام ناحیه می‌تواند **مطلق** (absolute) یا نسبی (relative) باشد. یک نام مطلق همیشه به نقطه ختم می‌شود (مانند *eng.sun.com.*)، در حالیکه نامهای نسبی چنین نیستند. نامهای نسبی بدون توجه به جایی که بکار رفته‌اند، معنی نمی‌دهند. در هر دو حالت، یک نام ناحیه به گرهی خاص در درخت نامها (و تمام گرههای ذیل آن) اشاره می‌کند.

کوچکی یا بزرگی حروف در نام ناحیه بی‌تأثیر است، بعبارت دیگر *edu* ، *Edu* و *EDU* همگی یک معنی می‌دهند. هر جزء از نام ناحیه حداکثر ۶۳ حرف، و کل مسیر نام ناحیه حداکثر ۲۵۵ حرف می‌توانند داشته باشند.

برای قرار دادن یک ناحیه در درخت نامهای ناحیه دو روش وجود دارد. برای مثال، ناحیهٔ *cs.yale.edu* می‌تواند در ذیل شاخهٔ کشور آمریکا (*us*) و بصورت *cs.yale.ct.us* هم ثبت شود. در کل، اغلب سازمانها و شرکتها در آمریکا ترجیح می‌دهند از ناحیه‌های عمومی استفاده کنند، در حالیکه در خارج از آمریکا بیشتر از نام کشور بعنوان ناحیهٔ سطح بالا استفاده می‌شود. هیچ منعی برای ثبت نام ناحیه در شاخه‌های متعدد وجود ندارد، با این حال چنین گرایشی بجز در شرکتهای چندملیتی (مانند شرکت سونی که ناحیه‌های *sony.com* و *sony.nl* را ثبت کرده) وجود ندارد.

هر ناحیه ناحیه‌های ذیل خود را کنترل می‌کند. برای مثال، کشور ژاپن دارای ناحیه‌های *ac.jp* و *co.jp* است، که بترتیب مشابه *edu* و *com* هستند؛ در حالیکه در هلند چنین تقسیم‌بندی وجود ندارد، و تمام ناحیه‌ها ذیل *nl* قرار دارند. برای مثال، ناحیه‌های زیر همگی دانشکده‌های کامپیوتر در کشور مربوطه هستند:

۱. cs.yale.edu (دانشگاه ییل، ایالات متحدهٔ آمریکا)

۲. cs.vu.nl (دانشگاه فریژه، هلند)

۳. cs.keio.ac.jp (دانشگاه کی‌یو، ژاپن)

برای ایجاد یک زیرناحیه، مجوز ناحیهٔ بالاتر مورد نیاز است. برای مثال اگر گروه VLSI در دانشکدهٔ کامپیوتر دانشگاه ییل تأسیس شود، و بخواهد به نام *vlsi.cs.yale.edu* شناخته شود، باید مجوزهای لازم را از مسئول *cs.yale.edu* کسب کند. بهمین ترتیب اگر دانشگاه جدیدی، مثلاً دانشگاه شمالی داکوتای جنوبی، تأسیس شود، و بخواهد ناحیهٔ *unsd.edu* را برای خود ثبت کند، باید هماهنگیهای لازم را با مسئول ناحیهٔ *edu* بعمل آورد. قرار دادن مسئولیت زیرناحیه‌ها بر عهدهٔ ناحیهٔ بالاتر باعث می‌شود تا هیچ دو ناحیه‌ای همنام نشوند. همین که یک ناحیه ایجاد شد (مانند، *unsd.edu*)، دیگر می‌تواند بدون نظارت ناحیه‌های بالاتر به ایجاد زیرناحیه‌های دلخواه خود (مثلاً *cs.unsd.edu*) بپردازد.

نامگذاری ناحیه‌ها امری سازمانی است نه فیزیکی. برای مثال، دانشکده‌های کامپیوتر و مهندسی برق یک دانشگاه می‌توانند ناحیه‌های کاملاً مستقلی داشته باشند، حتی اگر در یک ساختمان باشند و از شبکهٔ LAN واحدی استفاده کنند. از طرف دیگر، بخشهای مختلف یک دانشکده به یک ناحیه تعلق دارند، حتی اگر از نظر فیزیکی از هم جدا باشند.

۷-۱-۲ رکوردهای منابع

هر ناحیه، خواه ناحیه‌ای سطح بالا یا ناحیه‌ای با یک کامپیوتر، دارای تعدادی رکوردمنابع (resource record) است. برای یک کامپیوتر، متداولترین رکورد منبع آدرس IP آن است، اما انواع دیگری از رکوردهای منابع می‌تواند وجود داشته باشد. وقتی یک تبدیل‌کننده نام ناحیه را به DNS می‌دهد، چیزی که دریافت می‌کند تمام رکوردهای منابع وابسته به آن نام است. بنابراین، اصلی‌ترین وظیفهٔ یک DNS تبدیل نام ناحیه به رکوردهای منابع است.

هر رکورد منبع پنج بخش دارد. با اینکه رکوردهای منابع را می‌توان برای کارایی بهتر بصورت باینری در آورد، ولی در اغلب مواقع این رکوردها بصورت متنی (ASCII) - یک رکورد در هر خط - نگهداری می‌شوند. فرمت یک رکورد منبع مانند زیر است:

```
Domain_name     Time_to_live     Class     Type     Value
```

که در آن *Domain_name* نام ناحیه‌ایست که این رکورد متعلق به آن است. معمولاً هر ناحیه تعداد زیادی رکورد دارد، و در هر پایگاه داده اطلاعات چندین ناحیه نگهداری می‌شود. در نتیجه این فیلد کلید اصلی جستجو در پایگاه دادهٔ DNS است. (ترتیب قرار گرفتن رکوردها در پایگاه داده اهمیتی ندارد.)

فیلد *Time_to_live* مشخص می‌کند که این رکورد چقدر دوام می‌آورد. این فیلد در رکوردهای با دوام مقدار زیادی دارد، مثلاً ۸۶۴۰۰ (تعداد ثانیه‌های یک شبانه‌روز)؛ و بر عکس، اطلاعات کم دوام عمر کوتاهی دارد، مثلاً ۶۰ (یک دقیقه). در بحث حافظهٔ نهان به این موضوع برخواهیم گشت.

فیلد سوم هر رکورد منبع *Class* است. برای اطلاعات اینترنتی این فیلد همیشه *IN* است؛ برای اطلاعات غیراینترنتی از کُدهای دیگری هم می‌توان استفاده کرد (که البته بندرت دیده می‌شوند).

فیلد *Type* نوع رکورد منبع را مشخص می‌کند. مهمترین انواع رکوردهای منابع را در شکل ۷-۲ ملاحظه می‌کنید.

رکورد *SOA* نام منبع اصلی اطلاعات منطقه (zone) ، آدرس ایمیل سرپرست ناحیه، شماره سریال منحصر به فرد آن، و مشخصات دیگر ناحیه را مشخص می‌کند.

نوع	مفهوم	مقدار
SOA	Start of Authority	پارامترهای منطقه
A	IP address of a host	عدد صحیح ۳۲ بیتی
MX	Mail exchange	تقدم دریافت ایمیل
NS	Name Server	نام سرویس دهنده ناحیه
CNAME	Canonical name	نام ناحیه
PTR	Pointer	نام مستعار برای آدرس IP
HINFO	Host description	مشخصات CPU و سیستم عامل
TXT	Text	متن تسفیر نشده

شکل ۷-۲. انواع رکوردهای منابع اصلی DNS برای IPv4 .

مهمترین نوع رکورد منبع، رکورد *A* (آدرس) است. هر رکورد *A* یک آدرس IP ۳۲بیتی را در خود نگه می‌دارد. هر کامپیوتر اینترنت باید حداقل یک آدرس IP داشته باشد، تا کامپیوترهای دیگر بتوانند با آن تماس بگیرند. برخی از کامپیوترها دو یا چند آدرس IP دارند، که برای هر آدرس IP چنین کامپیوتری باید یک رکورد *A* وجود داشته باشد. DNS را می‌توان بگونه‌ای پیکربندی کرد که در میان این رکوردها بچرخد، یعنی برای اولین درخواست اولین رکورد را برگرداند، برای درخواست دوم دومین رکورد را، و الی آخر.

رکورد مهم بعدی رکورد *MX* است. این رکورد آدرس سرویس‌دهنده پُست الکترونیک (ایمیل) ناحیه را مشخص می‌کند. اختصاص یک نوع رکورد خاص به سرویس‌دهنده پست الکترونیک بدین خاطر است که تمام کامپیوترها چنین قابلیتی (دریافت ایمیل) ندارند. برای مثال، اگر کسی بخواهد به *bill@microsoft.com* ایمیل بفرستد، باید آدرس سرویس‌دهنده پست الکترونیک *microsoft.com* را پیدا کند. این اطلاعات را رکورد *MX* عرضه می‌کند.

رکورد *NS* **سرویس‌دهنده‌نام** (name server) را مشخص می‌کند. برای مثال، معمولاً هر پایگاه دادهٔ DNS یک رکورد NS برای ناحیه‌های سطح بالا دارد. (باز هم به این موضوع برمی‌گردیم.)

از رکورد *CNAME* می‌توان برای ایجاد **نامهای‌مستعار** (alias) استفاده کرد. برای مثال، فرض کنید دوستی بنام پاول در دانشکدهٔ مهندسی کامپیوتر دانشگاه MIT دارید، و می‌خواهید برای وی ایمیل بفرستید. فقط می‌دانید که نام وی در شبکهٔ این دانشگاه *paul* است، اما آدرس ایمیل کامل او را ندارید. حدس می‌زنید که ناحیهٔ دانشکدهٔ مزبور *cs.mit.edu* باشد، اما مسئولین این دانشکده (شاید از روی کج‌سلیقگی) نام *lcs.mit.edu* را برای ناحیهٔ خود را انتخاب کرده‌اند. اگر ایمیلی به آدرس *paul@cs.mit.edu* بفرستید، مسلماً برگشت خواهد خورد؛ ولی اگر مسئولین این دانشکده کمی هوشیاری بخرج دهند، با تعریف یک نام مستعار می‌توانند تا حدی اشتباه خود را جبران کنند - برای این منظور می‌توان از یک رکورد *CNAME* مانند زیر استفاده کرد:

```
cs.mit.edu          86400          IN          CNAME       lcs.mit.edu
```

رکورد *PTR* هم، مانند *CNAME* ، به یک نام دیگر اشاره می‌کند. اما بر خلاف *CNAME* (که در واقع یک ماکرو است)، رکورد *PTR* یک نوع دادهٔ معمولی DNS است که تفسیر آن به محتویات این رکورد بستگی دارد. در عمل، تقریباً همیشه از این رکورد برای **جستجوی‌معکوس** (reverse lookup) - تبدیل آدرس IP به نام ماشین - استفاده می‌شود.

رکورد *HINFO* اطلاعات مربوط به نوع ماشین و سیستم عامل آنرا برمی‌گرداند. از رکورد *TXT* هم می‌توان

برای برگرداندن اطلاعات متنی اضافی به کاربران استفاده کرد. رکوردهای *HINFO* و *TXT* فقط برای راحتی کاربران تعبیه شده‌اند، و الزامی نیستند. اغلب اوقات چنین اطلاعاتی وجود ندارد، و اگر هم وجود داشته باشد، نمی‌توان به آن کاملاً اطمینان کرد.

آخرین فیلد رکورد منبع، فیلد *Value* (مقدار) است. این فیلد می‌تواند یک عدد، نام ناحیه، یا یک رشتهٔ متنی باشد. طرز وارد کردن این مقدار به نوع آن بستگی دارد (در شکل ۷-۲ توضیح کوتاهی دربارهٔ نوع هر فیلد آورده شده است).

برای آشنایی بیشتر با اطلاعاتی که در پایگاه دادهٔ DNS یک ناحیه می‌توان یافت، شکل ۷-۳ را ببینید. در این شکل قسمتی از پایگاه دادهٔ نیمه‌فرضی ناحیه‌ای بنام *cs.vu.nl* (شکل ۷-۱) نشان داده شده است. در این پایگاه داده هفت نوع رکورد منبع وجود دارد.

اولین خط غیرتوضیحی شکل ۷-۳ مقداری اطلاعات اولیه دربارهٔ ناحیهٔ *cs.vu.nl* می‌دهد، که فعلاً با آنها کاری نداریم. دو خط بعدی مقداری اطلاعات متنی دربارهٔ این ناحیه (و محل آن) می‌دهند. پس از آن دو کامپیوتری که مسئول دریافت ایمیل‌های ناحیهٔ *cs.vu.nl* هستند، مشخص شده‌اند. اولین کامپیوتری که ایمیل‌ها باید به آن فرستاده شوند، *zephyr* نام دارد؛ و اگر *zaphyr* جواب نداد، نوبت به *top* می‌رسد.

بعد از یک خط خالی (که فقط برای خواناتر کردن فایل است)، رکوردهایی آمده‌اند که می‌گویند *flits* یک کامپیوتر Sun با سیستم عامل UNIX است، و دو آدرس IP دارد (130.37.16.112 و 192.31.231.165). پس از آن سه ماشین برای دریافت ایمیل‌های *flits.cs.vu.nl* مشخص شده است: اولین آنها طبیعتاً خود *flits* است، ولی

```
; Authoritative data for cs.vu.nl
cs.vu.nl.        86400  IN  SOA    star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.        86400  IN  TXT    "Divisie Wiskunde en Informatica."
cs.vu.nl.        86400  IN  TXT    "Vrije Universiteit Amsterdam."
cs.vu.nl.        86400  IN  MX     1 zephyr.cs.vu.nl.
cs.vu.nl.        86400  IN  MX     2 top.cs.vu.nl.

flits.cs.vu.nl.  86400  IN  HINFO  Sun Unix
flits.cs.vu.nl.  86400  IN  A      130.37.16.112
flits.cs.vu.nl.  86400  IN  A      192.31.231.165
flits.cs.vu.nl.  86400  IN  MX     1 flits.cs.vu.nl.
flits.cs.vu.nl.  86400  IN  MX     2 zephyr.cs.vu.nl.
flits.cs.vu.nl.  86400  IN  MX     3 top.cs.vu.nl.
www.cs.vu.nl.    86400  IN  CNAME  star.cs.vu.nl
ftp.cs.vu.nl.    86400  IN  CNAME  zephyr.cs.vu.nl

rowboat                 IN  A      130.37.56.201
                        IN  MX     1 rowboat
                        IN  MX     2 zephyr
                        IN  HINFO  Sun Unix

little-sister           IN  A      130.37.62.23
                        IN  HINFO  Mac MacOS

laserjet                IN  A      192.31.231.216
                        IN  HINFO  "HP Laserjet IIISi" Proprietary
```

شکل ۷-۳. قسمتی از پایگاه دادهٔ DNS در ناحیهٔ *cs.vu.nl* .

اگر این ماشین خاموش بود، *zephyr* و *top* گزینه‌های بعدی خواهند بود. بعد از آن یک نام مستعار برای ماشین *star.cs.vu.nl* تعریف شده است: *www.cs.vu.nl* . با تعریف این نام مستعار، کاربران می‌توانند بدون دغدغهٔ تغییر آدرس صفحهٔ وب *cs.vu.nl* به آن مراجعه کنند. همین کار برای *ftp.cs.vu.nl* نیز انجام شده است.

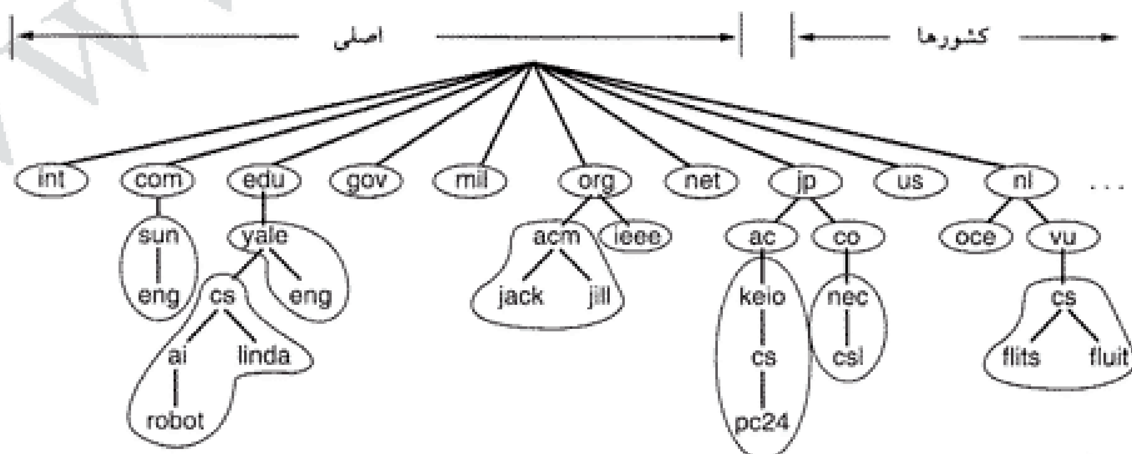
در چهار خط بعدی رکوردهای منبع کامپیوتری بنام *rowboat.cs.vu.nl* تعریف شده‌اند. این اطلاعات شامل آدرس IP ، محل دریافت ایمیل (اولیه و ثانویه)، و اطلاعاتی دربارهٔ خود ماشین است. پس از آن یک کامپیوتر MAC (بدون قابلیت دریافت ایمیل)، و بدنبال آن یک چاپگر لیزری که به اینترنت متصل است، تعریف شده‌اند.

چیزی که در اینجا نشان داده نشده (و در واقع در این فایل هم نیست)، آدرس IP ناحیه‌های سطح بالا است. از آنجائیکه این کار در حوزهٔ مسئولیت ناحیهٔ *cs.vu.nl* نیست، در این فایل هم رکوردی برای آن وجود ندارد. این قبیل اطلاعات را کامپیوترهایی بنام **سرویس‌دهنده ریشه** (root server) ارائه می‌کنند، که با هر بار اجرای سرویس‌دهنده DNS آدرس آنها در حافظه حافظهٔ نهان DNS بار می‌شود. تعداد سرویس‌دهنده‌های ریشه در حدود ۱۲ تاست که در سراسر دنیا پراکنده‌اند، و آدرس IP تمام ناحیه‌های سطح بالا را دارند. بنابراین، اگر ماشینی آدرس IP حداقل یکی از این سرویس‌دهنده‌های ریشه را داشته باشد، می‌تواند نام هر ناحیه‌ای را پیدا کند.

۷-۱-۳ سرویس‌دهنده نام

از نظر تئوری، برای نگهداری تمام اطلاعات DNS و پاسخ دادن به درخواست‌ها یک سرویس‌دهنده DNS کافیست. اما در عمل، بار کاری چنین کامپیوتری آنقدر سنگین خواهد شد که عملاً آنرا بلااستفاده می‌کند. علاوه بر آن، اگر این کامپیوتر از کار بیفتد، تمام اینترنت هم با آن به خواب خواهد رفت.

برای اجتناب از چنین وضعیتی، فضای نام DNS به چندین **منطقه** (zone) با مرزهای مشخص و غیرمشترک تقسیم شده است. در شکل ۷-۴ یکی از راههای تقسیم فضای نام شکل ۷-۱ را مشاهده می‌کنید. هر منطقه شامل بخشی از درخت DNS است، و سرویس‌دهنده‌های نام آنرا در خود نگه می‌دارد. معمولاً هر منطقه دارای یک **سرویس‌دهنده نام** (name server) اولیه است که اطلاعاتش را از فایلی روی دیسک خود می‌گیرد، و یک یا چند سرویس‌دهنده نام ثانویه نیز دارد که آنها اطلاعات خود را از سرویس‌دهنده نام اولیه می‌گیرند. برای بالا بردن ضریب اطمینان، می‌توان تعدادی از سرویس‌دهنده‌های نام یک منطقه را خارج از آن منطقه مستقر کرد.



شکل ۷-۴. فضای نام DNS به منطقه‌های مختلف تقسیم شده است.

تعیین مرزهای یک منطقه بر عهدهٔ سرپرست آن است. تصمیم‌گیری در این باره تا حد زیادی به تعداد سرویس‌دهنده‌های نام منطقه و محل استقرار آنها بستگی دارد. برای مثال، در شکل ۷-۴، دانشگاه ییل دارای یک

سرویس‌دهنده نام برای ناحیه‌های *yale.edu* و *eng.yale.edu* است، اما ناحیهٔ *cs.yale.edu* در منطقهٔ دیگری قرار دارد. چنین تصمیماتی بیشتر به تمایل ناحیه‌ها برای کنترل مستقیم منطقهٔ خود بستگی دارد. در مثال فوق، ناحیهٔ *cs.yale.edu* منطقه‌ای مستقل است، در حالیکه *eng.yale.edu* چنین نیست.

وقتی یک تبدیل‌کننده می‌خواهد آدرس ناحیه‌ای را بداند، ابتدا درخواست خود را به سرویس‌دهنده‌های نام محلی خود می‌دهد. اگر این ناحیه در محدودهٔ قانونی سرویس‌دهنده نام مزبور بود (مانند *ai.cs.yale.edu* که در قلمرو *cs.yale.edu* است)، سرویس‌دهنده نام رکوردهای منبع معتبر را به آن برمی‌گرداند. یک **رکورد معتبر** (authoritative record) رکوردی است که مستقیماً از سرپرست ناحیه منشأ می‌گیرد، و بنابراین همیشه صحیح و معتبر است (بر خلاف **رکوردهای ذخیره‌شده** – cached record – که تاریخ اعتبار آنها می‌تواند منقضی شده باشد).

ولی اگر آن ناحیه در قلمرو سرویس‌دهنده‌های محلی نباشد، سرویس‌دهنده نام این درخواست را به سرویس‌دهنده نام سطح بالای ناحیهٔ مزبور می‌فرستد. برای روشنتر شدن مطلب، به مثالی در شکل ۷–۵ توجه کنید. در اینجا یک تبدیل‌کننده در ناحیهٔ *flits.cs.vu.nl* می‌خواهد آدرس IP ماشین سرویس‌دهنده ناحیهٔ *linda.cs.yale.edu* را بداند. در مرحله ۱، این تبدیل‌کننده درخواست خود را به سرویس‌دهنده نام محلی، یعنی *cs.vu.nl*، می‌فرستد. این درخواست شامل نام ناحیهٔ موردنظر (رکوردهای نوع *A* و کلاس *IN*) می‌باشد.



شکل ۷–۵. مراحل جستجوی نام ناحیه.

اجازه دهید فرض کنیم سرویس‌دهنده نام محلی تا بحال چیزی از ناحیهٔ *linda.cs.yale.edu* نشنیده و دربارهٔ آن هیچ اطلاعاتی ندارد. این سرویس‌دهنده می‌تواند از همسایه‌های خود در این باره پرس و جو کند، ولی اگر آنها هم بی‌اطلاع بودند، یک بستهٔ UDP به سرویس‌دهنده نام *edu* یعنی *edu-server.net* (که آدرس آنرا در حافظه‌اش دارد) می‌فرستد (شکل ۷–۵ را ببینید). احتمال کمی هست که این سرویس‌دهنده آدرس *linda.cs.yale.edu* (یا حتی *cs.yale.edu*) را بداند، ولی حتماً بچه‌های خودش را می‌شناسد، پس درخواست را به سرویس‌دهنده نام *yale.edu* منتقل می‌کند (مرحلهٔ ۳). سرویس‌دهنده *yale.edu* هم درخواست را به *cs.yale.edu* هدایت می‌کند (مرحلهٔ ۴)، که باید اطلاعات معتبر را در اختیار داشته باشد. از آنجائیکه این مسیر از مشتریهای مختلفی عبور کرده، رکوردهای درخواستی هم باید از همان مسیر به *flits.cs.vu.nl* برگردند (مراحل ۵ تا ۸).

وقتی این رکوردها به *cs.vu.nl* رسید، در حافظهٔ نهان آنجا ذخیره می‌شود تا در دفعات بعد مورد استفاده قرار گیرد. ولی این اطلاعات معتبر نیستند، چون هر تغییری که در *cs.yale.edu* داده شود بطور خودکار در حافظهٔ نهان DNS هایی که این رکوردها در آنجا وجود دارد، پخش نخواهد شد. بهمین دلیل، آیتمهای حافظهٔ نهان نباید عمری طولانی داشته باشند؛ و علت قرار دادن فیلد *Time_to_live* در رکوردهای منبع نیز همین است. این فیلد به سرویس‌دهنده نام می‌گوید که تا چه مدتی می‌تواند رکورد را در حافظهٔ نهان خود نگه دارد. اگر یک ماشین IP خود را سالها حفظ می‌کند، نگه داشتن آن برای ۱ روز در حافظهٔ نهان DNS چندان غیرمنطقی نیست. اما اطلاعات ناپایدارتر را بهتر است بیش از چند ثانیه (یا حداکثر ۱ دقیقه) در حافظهٔ نهان نگه نداریم.

به پرس و جوهایی که به طریق بالا عمل می‌کنند، **جستجوی بازگشتی** (recursive query) می‌گویند، چون هر سرویس‌دهنده‌ای که اطلاعات خواسته شده را نداشته باشد، آنرا به سرویس‌دهنده بالاتر هدایت کرده و جواب را

باز می‌گرداند. روش جستجوی دیگری نیز وجود دارد: سرویس‌دهنده‌ای که اطلاعات درخواست شده را ندارد، خود به سرویس‌دهنده بالاتر مراجعه نمی‌کند، بلکه آدرس آنرا به درخواست کننده برمی‌گرداند. برخی از سرویس‌دهنده‌های DNS قادر به انجام جستجوی بازگشتی نیستند، و همیشه آدرس سرویس‌دهنده بعدی را برمی‌گردانند.

اگر یک مشتری DNS در زمان مقرر پاسخ خود را دریافت نکند، سراغ سرویس‌دهنده DNS بعدی خواهد رفت. این اتفاق بیشتر در مواردی رخ می‌دهد که سرویس‌دهنده DNS بدلایلی از مدار خارج شده باشد.

با اینکه DNS کار ساده‌ای انجام می‌دهد (تبدیل نام به آدرس IP)، اما کارکرد صحیح آن در اینترنت اهمیت حیاتی دارد. DNS هیچ کمکی برای یافتن افراد، منابع، سرویسها، اشیاء و چیزهایی از این قبیل نمی‌تواند بکند؛ برای این کارها سرویس دیگری بنام LDAP (پروتکل سبک‌دسترسی دایرکتوری – Light-weight Directory Access Protocol) تعریف شده است. این سرویس شکل ساده شدهٔ سرویس دایرکتوری OSI X.500 است، که در استاندارد RFC 2251 تشریح شده است. در LDAP (که می‌توان آنرا «دفتر تلفن اینترنتی» بشمار آورد)، اطلاعات بصورت درختی منظم شده‌اند، و امکانات فراوانی برای جستجو در این درخت تعبیه شده است. در این کتاب بیش از این دربارهٔ LDAP صحبت نخواهیم کرد؛ برای کسب اطلاعات بیشتر می‌توانید به (Weltman and Dahbura, 2000) مراجعه کنید.

۲ـ۷ پُست الکترونیک

بیش از دو دهه است که پُست الکترونیک، یا آنطور که هوادارانش می‌گویند ایمیل (e-mail)، در صحنه حضور دارد. تا سال ۱۹۹۰، این سرویس بیشتر در دانشگاهها و مراکز علمی وجود داشت، ولی وقتی در این سال بصورت سرویسی عمومی در آمد، با چنان سرعتی رشد کرد که در طی یک دهه تعداد نامه‌های الکترونیکی فرستاده شده از تعداد نامه‌های کاغذی فراتر رفت.

ایمیل، مانند سایر روشهای ارتباطی، دارای قواعد و شیوه‌های خاص خود است. جاذبهٔ ایمیل بسیار بالاست، بطوریکه حتی آنهائیکه بندرت نامه‌های معمولی می‌نویسند، در نوشتن نامه‌های الکترونیکی (حتی به مقامات رسمی و سطح بالا) تردیدی بخود راه نمی‌دهند.

نامه‌های الکترونیکی پُر از کلماتیست که قبلاً در هیچ کجا دیده نشده‌اند: BTW (By The Way – راستی)، ROTFL (Rolling On The Floor Laughing – از خـنده غش کـردم)، و IMHO (In My Humble Opinion – به نظر من ناقابل) از آن نمونه‌اند. بسیاری افراد نیز در ایمیلهای خود از علائم خاصی موسوم به خندانک (smiley) یا احساس‌نما (emoticon) استفاده می‌کنند. در شکل ۷-۶ تعدادی از معروفترین این خندانک‌ها (و معنای آنها) را می‌بینید. اگر می‌خواهید بهتر متوجه معنای این علائم شوید، کتاب را °۹۰ در جهت

مفهوم	خندانک	مفهوم	خندانک	مفهوم	خندانک
دماغ گنده	:+)	عمو لینکن	=|:-)	من خوشحالم	:-)
غبغب بزرگ	:-))	عمو سام	=):-)	من غمگین / ناراحتم	:-(
سیبیلو	:-{)	بابا نوئل	*<:-)	من بی تفاوتم	:-|
ژولیده مو	#:-)	کودن / احمق	<:-(___	من چشمک می زنم	;-)
عینکی	8-)	استرالیایی	(-:	من خمیازه می کشم	:-(O)
با هوش	C:-)	مرد فلکی	:-)X	حالم به هم خورد	:-(*)

شکل ۷-۶. چند خندانک. اینها جزء امتحان نهایی نیستند :-)

عقربه‌های ساعت بچرخانید. برای دیدن تعداد زیادی از این قبیل خنداننک‌ها به (Sanderson and Dougherty, 1993) مراجعه کنید.

اولین سیستم ایمیل فقط یک پروتکل سادهٔ انتقال فایل (file transfer) بود، که آدرس گیرنده در خط اول پیام (فایل) نوشته می‌شد. با گذشت زمان محدودیت‌های این روش آشکارتر شد، که برخی از آنها عبارت بودند از:

۱. فرستادن یک پیام به چند نفر مشکل بود. این اشکال بیشتر مدیران را آزار می‌داد، چون آنها میل داشتند پیامهای خود را به تمام افراد زیردست خود بفرستند.
۲. پیامها هیچگونه ساختار داخلی نداشتند، و بهمین دلیل پردازش کامپیوتری آنها مشکل بود. برای مثال، اگر پیامی از طرف یک شخص واسطه هدایت یا فرستاده می‌شد، استخراج قسمت هدایت شده مشکل بود.
۳. فرستندهٔ نامه هرگز نمی‌توانست بداند پیامش به گیرنده رسیده یا نه.
۴. اگر کسی قصد داشت برای مدتی به مرخصی برود و می‌خواست در این مدت نامه‌های وارده به دست منشی‌اش برسد، کار ساده‌ای نبود.
۵. واسط کاربر (جایی که نامه را می‌نوشت) با قسمت ارسال نامه یکپارچه نبود. کاربر باید ابتدا نامه را می‌نوشت، و برای ارسال آن برنامهٔ ادیتور را ترک می‌کرد، و به قسمت انتقال فایل می‌رفت.
۶. نامه‌ها فقط متن بود؛ ارسال تصویر، طرح، صدا و مانند آنها ممکن نبود.

بتدریج سیستمهای ایمیل بهتری عرضه شد. در سال ۱۹۸۲، آرپانت سیستم ایمیل پیشنهادی خود را در RFC 821 (پروتکل انتقال) و RFC 822 (فرمت پیام) ارائه کرد. این پیشنهادها با تغییراتی اندک با عنوان RFC 2821 و RFC 2822 به استاندارد اینترنت تبدیل شد - اما هنوز هم استاندارد ایمیل اینترنت را با نام RFC 822 می‌شناسند.

در ۱۹۸۴، CCITT توصیه‌ای بنام X.400 ارائه کرد. بعد از دو دهه، اغلب سیستمهای ایمیل همچنان به RFC 822 پایبند هستند، و X.400 عملاً کنار گذاشته شده است. این که چگونه سیستمی به عظمت X.400 که تمام مقامات رسمی استاندارد، شرکتهای مخابرات سراسر دنیا، دولتها و بسیاری از صنایع کامپیوتری پشتیبان آن بودند، مغلوب سیستمی که چند دانشجوی کامپیوتر آنرا نوشته‌اند، می‌شود بیشتر به داستان داوود و گولیات شبیه است.

علت موفقیت RFC 822 خوبی آن نبود، بلکه این X.400 بود که چنان پیچیده و بد طراحی شده بود که پیاده‌سازی آن را عملاً غیرممکن می‌کرد. انتخاب بین یک سیستم ساده ولی کاری (مانند RFC 822) و سیستمی فوق‌العاده جالب که در عمل کار نمی‌کرد (مانند X.400) چندان دشوار نبود. این عبرت تاریخ است.

۷-۲-۱ معماری و سرویسها

در این قسمت خواهید دید که یک سیستم ایمیل چه کاری می‌تواند انجام دهد، و سازماندهی آن چگونه است. هر سیستم ایمیل دارای دو زیرسیستم است: **عامل کاربر** (user agent) ، که به افراد اجازه می‌دهد پیامهای خود را بفرستند و پیامهای رسیده را بخوانند، و **عامل انتقال پیام** (message transfer agent) ، که پیامها را به دست گیرنده می‌رساند. عامل کاربر برنامه‌ایست (با ظاهر معمولی) روی کامپیوتر محلی کاربر، که با سیستم ایمیل بر هم کنش دارد. در حالیکه عامل انتقال پیام معمولاً یک **سرویس** (service یا daemon) است که در پس‌زمینه اجرا می‌شود، و وظیفهٔ آن انتقال پیام در سیستم ایمیل است.

یک سیستم ایمیل، معمولاً، پنج کارکرد اصلی دارد، که آنها را در زیر توضیح می‌دهیم.

تصنیف: نوشتن پیام و جواب آن. با اینکه از هر ادیتوری می‌توان برای نوشتن پیامها استفاده کرد، ولی اغلب سیستمهای ایمیل دارای ادیتور خاص خود هستند که بسیاری از کارها (از جمله نوشتن آدرس، و سرآیند ایمیل) را بطور خودکار انجام می‌دهند. برای مثال، وقتی می‌خواهید به یک نامه جواب بدهید، سیستم ایمیل می‌تواند آدرس فرستندهٔ نامه را بطور خودکار استخراج کرده و در فیلد گیرندهٔ جوابیه (reply) قرار دهد.

انتقال: فرستادن پیام از فرستنده به گیرنده. این فرآیند سه مرحله دارد: تماس با ماشین گیرنده (یا یک ماشین واسط)، فرستادن پیام، و قطع ارتباط. سیستم ایمیل این کارها را بطور خودکار و بدون دخالت کاربر انجام می‌دهد.

گزارش‌دهی: مطلع کردن کاربر از سرنوشت پیام فرستاده شده. نامه تحویل شد؟ گیرنده آنرا قبول نکرد؟ در راه گم شد؟ در برخی مواقع اطمینان از رسیدن نامه بدست گیرنده اهمیت حیاتی و تبعات قانونی دارد (مانند احضاریه‌های دادگاه).

نمایش: پیامهای رسیده باید بگونه‌ای مناسب در معرض دید کاربر قرار گیرند، تا وی بتواند براحتی آنها را بخواند. گاهی لازم است برای خواندن محتویات برخی نامه‌ها (مانند نامه‌هایی که پیوست صوتی یا تصویری دارند) از برنامه‌های کمکی استفاده شود. برخی از سیستمهای ایمیل نیز نامه‌ها را بگونه‌ای خاص فرمت کرده و نمایش می‌دهند.

بایگانی: نگهداری نامه‌های رسیده. سرنوشت پیامهای رسیده متفاوت است: برخی پیامها حتی قبل از خوانده شدن دور انداخته می‌شوند، برخی فقط یک بار ارزش خواندن دارند، و برخی دیگر را باید حتماً ذخیره کرد. یک سیستم ایمیل باید بتواند نامه‌ها را به انحاء مختلف پردازش کند.

علاوه بر این سرویسهای اصلی، برخی از سیستمهای ایمیل (مخصوصاً سیستمهای رسمی) دارای ویژگیهای پیشرفتهٔ دیگری نیز هستند، که در زیر به برخی از آنها اشاره می‌کنیم.

وقتی یک فرد از محلی نقل مکان می‌کند (یا برای مدتی به مأموریت می‌رود)، باید بتوان پیامهای وی را به محل جدید **هدایت‌کرد** (forward) . سیستم ایمیل باید بتواند این کار را بصورت خودکار انجام دهد.

در اکثر سیستمها کاربران اجازه دارند برای ذخیره کردن پیامهای خود **صندوق‌پستی** (mailbox) داشته باشند. سیستم ایمیل باید فرمانهایی برای ایجاد، مدیریت و یا از بین بردن این صندوق‌ها داشته باشد.

اغلب مدیران نیاز دارند تا یک پیام را به افراد متعددی (کارمندان، مشتریان، یا شرکتهای طرف قرارداد) بفرستند. از اینجا بود که ایدهٔ **لیست‌پستی** (mailing list) - که در واقع لیستی از آدرسهای ایمیل است - پیدا شد. وقتی پیامی به یک لیست پستی فرستاده می‌شود، تمام افراد لیست کپی‌های کاملاً یکسانی از آن پیام دریافت خواهند کرد.

ویژگیهای دیگر عبارتند از: کپی (CC) ، کپی ناشناس (BCC) ، ایمیل با اولویت زیاد، ایمیل سرّی (رمز شده)، گیرندهٔ جانشین (وقتی گیرندهٔ اصلی در دسترس نبود)، و تحویل نامه‌های رئیس به منشی.

امروزه ایمیل کاربرد گسترده‌ای برای ارتباطات داخلی شرکتها و سازمانها دارد. ایمیل می‌تواند گروه گسترده‌ای از افراد (حتی آنهایی که بسیار از یکدیگر دور هستند) را در یک پروژه گرد آورد. ایمیل با حذف اکثر تمایزها (مانند مقام، سن، و جنسیت) باعث تمرکز روی اهداف می‌شود. با ایمیل، ایدهٔ درخشان یک کارآموز ساده اهمیت فزونتری نسبت به ایده‌های (اکثراً احمقانهٔ) مدیر عامل خواهد یافت.

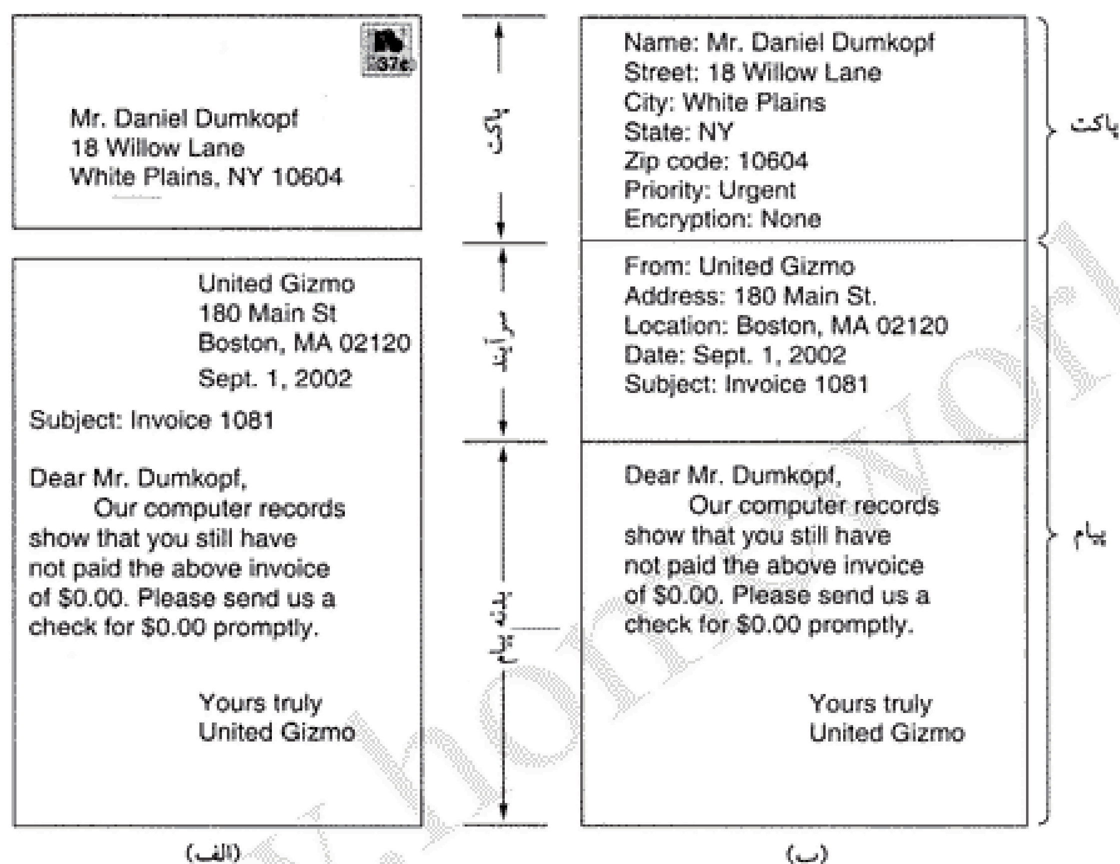
ایدهٔ کلیدی در سیستم ایمیل تمایز بین پاکت و محتویات نامه است. پاکت نامه پیام را در خود جای می‌دهد، و شامل اطلاعاتی از قبیل آدرس گیرنده، اولویت و سطح امنیتی آن می‌شود، که معمولاً ارتباطی با محتویات نامه ندارند. عامل انتقال پیام از اطلاعات این پاکت برای جابجایی صحیح پیام استفاده می‌کند (درست مثل ادارهٔ پُست).

محتویات پاکت دو بخش دارد: **سرآیند** (header) و **بدنه** (body) . سرآیند شامل اطلاعات کنترلیست که عامل کاربر از آنها برای کار خود استفاده می‌کند. بدنه بخشی از پیام است که به گیرنده مربوط می‌شود. در شکل ۷-۷ رابطهٔ پاکت و پیام نشان داده شده است.

۷-۲-۲ عامل کاربر

همانطور که گفتیم، سیستمهای ایمیل دو بخش اصلی دارند: عامل کاربر، و عامل انتقال پیام. در این قسمت بخش

اول (یعنی، عامل کاربر) را بررسی خواهیم کرد. عامل کاربر یک برنامهٔ معمولیست (و گاهی بـه آن نـامه‌خوان – mail reader – نیز می‌گویند)، که می‌تواند در نوشتن پیامها، خواندن نامه‌های رسیده، پاسخ به آنها و کارهایی از این قبیل به کاربر کمک کند. طیف وسیع و متنوعی از برنامه‌های عامل کاربر وجود دارد، اما کارکرد اصلی آنها بسیار شبیه یکدیگر است.



شکل ۷-۷. پاکت و پیام در (الف) پُست کاغذی، (ب) پُست الکترونیک.

فرستادن ایمیل

برای فرستادن یک ایمیل، کاربر باید ابتدا متن نامه را نوشته و آدرس گیرنده را هم مشخص کند. برای نوشتن نامه می‌توان از ادیتورها یا واژه‌پردازهای مستقل، و یا ادیتوری که به همراه عامل کاربر مـی‌آید، اسـتفاده کـرد. آدرس گیرنده باید با فرمتی باشد که برای عامل کاربر آشناست؛ بسیاری از آنها آدرسها را با فرمت *user@dns-address* می‌پذیرند. با فرمت نامهای DNS در ابتدای همین فصل آشنا شدید.

غیر از آدرسهای DNS فرمتهای دیگری هم برای آدرسهای ایمیل وجود دارد، که بویژه نوع X.400 آن قابل توجه است. آدرسهای X.400 تفاوت قابل توجهی با آدرسهای DNS دارند. هر آدرس X.400 مجموعه ایست از زوجهای *attribute* = *value* که با / از هم جدا می‌شوند:

/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

در این آدرس بترتیب از چپ براست کشور (C) ، ایالت (ST) ، محل (L) ، آدرس (PA) ، و نام شـخص (CN) نوشته می‌شود. مشخصات دیگری (مانند شرکت و شغل) نیز وجود دارد، که با استفاده از آنها می‌توانید پیام خود را حتی به فردی که آدرس ایمیل او را نمی‌دانید، بفرستید. با اینکه آدرسهای X.400 بسیار غریب‌تر از نامهای

DNS هستند، اما اغلب سیستمهای ایمیل اجازه می‌دهند تا هر آدرس یک نام مستعار ساده (موسوم به nickname) داشته باشد. بدین‌ترتیب دیگر لازم نیست کاربر آدرس ایمیل (حتی X.400) را بطور کامل وارد کند.

اغلب سیستمهای ایمیل از لیست پُستی پشتیبانی می‌کنند، بنابراین می‌توان یک نامه را یکباره برای تعداد زیادی از افراد فرستاد. اگر لیست پستی بصورت محلی نگهداری شود، ایمیل در همان مبدأ بین افراد آن لیست توزیع خواهد شد. ولی اگر لیست پستی در محل دیگری نگهداری شود، پیام در آنجا پخش می‌شود. برای مثال، اگر گروهی از طرفداران حیات وحش یک لیست پستی بنام *birders* در دانشگاه آریزونا داشته باشند *(birders@meadowlark.arizona.edu)* ، نامه‌ای که به این آدرس فرستاده شده باشد، فقط بعد از رسیدن به دانشگاه آریزونا بین افراد آن توزیع خواهد شد. آنهایی که در یک لیست پستی قرار دارند، بهیچوجه متوجه این مطلب نخواهند شد؛ آنها هم مانند سایر افراد پیامهایی از یک فرستندهٔ مشخص دریافت می‌کنند.

خواندن ایمیل

معمولاً، وقتی عامل کاربر کار خود را شروع می‌کند، قبل از هر چیز صندوق پستی کاربر را چک کرده و نامه‌های رسیده را از آنجا برمی‌دارد. پس از آن، تعداد نامه‌های رسیده را به کاربر اعلام کرده، و احتمالاً آنها را بصورت خلاصه (شامل اطلاعاتی از قبیل فرستنده، تاریخ دریافت، و موضوع نامه) به کاربر نمایش می‌دهد، و سپس منتظر اقدام بعدی کاربر می‌ماند.

بعنوان مثال، یک سناریوی ساده را در زیر بررسی می‌کنیم. بعد از شروع برنامهٔ عامل کاربر، اولین اقدام معمولاً نمایش خلاصه‌ای از نامه‌های رسیده است. این خلاصه می‌تواند چیزی شبیه شکل ۷-۸ باشد: هر خط مشخصات یک پیام را نشان می‌دهد. در این مثال، هشت پیام در صندوق پستی وجود دارد.

#	پرچم	بایت	فرستنده	موضوع
1	K	1030	asw	Changes to MINIX
2	KA	6348	trudy	Not all Trudys are nasty
3	K F	4519	Amy N. Wong	Request for information
4		1236	bal	Bioinformatics
5		104110	kaashoek	Material on peer-to-peer
6		1223	Frank	Re: Will you review a grant proposal
7		3110	guido	Our paper has been accepted
8		1204	dmr	Re: My student's visit

شکل ۷-۸ نمایش محتویات صندوق پستی.

در هر خط اطلاعات مختلفی دیده می‌شود. در یک سیستم ایمیل ساده این اطلاعات ثابت است، ولی در سیستمهای پیچیده‌تر کاربر می‌تواند نحوهٔ نمایش اطلاعات را بنا به میل خود تغییر دهد (و این تنظیمات را در فایلی بنام **پروفایل کاربر** – user profile – ذخیره کند). در مثال بالا، اولین فیلد شمارهٔ پیام است. فیلد دوم، *Flags* ، مقادیر مختلفی می‌تواند بگیرد: *K* یعنی این پیام آرشیوی است (جدید نیست و قبلاً خوانده شده)؛ *A* یعنی به این پیام پاسخ داده شده است؛ *F* یعنی این پیام به فرد دیگری هدایت شده است (forward) . پرچمهای دیگری نیز در این فیلد می‌تواند وجود داشته باشد، که هر کدام معنای خاص خود را دارند.

فیلد سوم طول پیام، و فیلد چهارم فرستندهٔ آنرا نشان می‌دهند. از آنجائیکه این فیلد از نامهٔ رسیده استخراج

می‌شود، محتویات آن به نامهٔ فرستاده شده بستگی دارد. و بالاخره، در فیلد **موضوع** خلاصه‌ای از محتویات نامه را می‌بینید. سعی کنید همیشه نامه‌هایتان «موضوع» داشته باشد، چون تجربه نشان داده که نامه‌های بدون موضوع براحتی نادیده گرفته می‌شوند.

بعد از خواندن خلاصهٔ نامه، کاربر می‌توان اقدامات مختلفی روی آن انجام دهد: نامه را باز کند، آنرا حذف کند، به نامه جواب دهد، آنرا برای کس دیگری بفرستد، و مانند آن. برای هر یک از این اعمال، فرمان خاصی در برنامهٔ عامل کاربر وجود دارد.

سیستمهای ایمیل امروزی خیلی بیش از انتقال فایل ساده‌اند. مدیریت حجم زیادی از نامه‌ها با برنامه‌های امروزی کار چندان دشواری نیست؛ و برای افرادی که در سال هزاران ایمیل رد و بدل می‌کنند، این مزیت کوچکی نیست.

۳-۲-۷ فرمت پیامها

اجازه دهید کمی هم دربارهٔ فرمت پیامهای ایمیل صحبت کنیم. ابتدا به ایمیلهای متنی ساده با فرمت RFC 822 می‌پردازیم، و پس از آن دربارهٔ الحاقات چندرسانه‌ای در RFC 822 توضیح خواهیم داد.

RFC 822

هر پیام یک پاکت ساده (که مشخصات آن در RFC 821 آمده) دارد، بعلاوهٔ تعدادی سرآیند، یک خط خالی، و بعد از آن متن یا بدنهٔ پیام. هر فیلد سرآیند عبارتست از یک خط متن ASCII مشتمل بر: نام فیلد، علامت : ، و مقدار فیلد (که البته برخی از فیلدها می‌توانند مقدار نداشته باشند). در استاندارد RFC 822 که بیش از دو دهه از عمر آن می‌گذرد، تمایز آشکاری بین فیلدهای پاکت نامه و فیلدهای سرآیند وجود ندارد. با اینکه در RFC 2822 این مشکل مرتفع شده، ولی بعلت رواج گستردهٔ آن در عمل چنین اتفاقی نیفتاده است. در حالت عادی، عامل انتقال پیام با استخراج اطلاعات از نامه‌ای که از عامل کاربر دریافت می‌کند، پاکت نامه را می‌سازد، و بهمین دلیل پاکت و نامه با هم مخلوط می‌شوند.

مهمترین فیلدهای سرآیند که نقش مهمی در انتقال پیام دارند، در شکل ۷-۹ نشان داده شده است. در فیلد *:TO* آدرس DNS گیرندهٔ اصلی پیام نوشته می‌شود. نوشتن چندین گیرنده در این فیلد مجاز است. در فیلد *:Cc* آدرس گیرنده‌های ثانویهٔ پیام نوشته می‌شود. از نظر تحویل پیام در سیستم ایمیل، تفاوتی بین گیرنده‌های اصلی و ثانویه وجود ندارد؛ این فیلد بیشتر برای انسانها مهم است تا برای ماشین. اصطلاح *Cc* (کپی کربنی) قدری قدیمی است، چون در کامپیوترها چیزی بنام کاغذ کپی وجود ندارد، اما این اصطلاح دیگر کاملاً جا افتاده است. فیلد *:Bcc* (کپی کربنی ناپیدا) شبیه *:Cc* است، با این تفاوت که این فیلد در نامه‌های کپی شده حذف می‌شود، و گیرندهٔ نامه از هویت سایر گیرنده‌ها (و حتی وجود چنین گیرنده‌هایی) مطلع نخواهد شد.

سرآیند	مفهوم
To:	آدرس ایمیل گیرنده های اصلی
Cc:	آدرس ایمیل گیرنده های ثانویه (کپی)
Bcc:	آدرس ایمیل گیرنده های کپی‌های ناشناس
From:	فرستنده پیام
Sender:	آدرس ایمیل فرستنده
Received:	هر عامل انتقال واسطه در بین راه مشخصات خود را اضافه می کند
Return-Path:	می توان از آن برای مشخص کردن مسیر برگشت به فرستنده استفاده کرد

شکل ۷-۹. فیلدهای سرآیند RFC 822 برای انتقال پیام.

دو فیلد بعدی، یعنی :From و :Sender ، بترتیب نویسنده و فرستندهٔ نامه را مشخص می‌کنند. این دو الزاماً یکی نیستند (ولی اغلب چنین است). برای مثال، نویسندهٔ نامه می‌تواند مدیر عامل باشد (:From) ، ولی منشی شرکت آنرا بفرستد (:Sender) . فیلد :From حتماً باید پُر شود، ولی فیلد :Sender (اگر با :From یکی باشد) می‌تواند خالی رها شود. اگر احتمال می‌دهید نامه بدست گیرنده نمی‌رسد و برگشت می‌خورد، حتماً فیلد :Sender را پُر کنید، چون نامه‌های غیرقابل‌تحویل به این آدرس برگشت داده می‌شوند.

وقتی یک نامه از واسطه‌های مختلفی عبور می‌کند تا به دست گیرنده برسد، نام هر واسطه در یک فیلد :Received جداگانه نوشته می‌شود. در این فیلد نام عامل گیرنده، تاریخ و زمان دریافت پیام، و اطلاعات دیگر ثبت می‌شود. از این اطلاعات می‌توان برای رفع اشکالاتی که در طی تحویل نامه‌ها پیش می‌آید، استفاده کرد.

فیلد :Return-Path که توسط آخرین عامل انتقال پیام اضافه می‌شود، نشان می‌دهد که مسیر برگشت به فرستنده چگونه است. از نظر تئوری این اطلاعات باید شامل تمام سرآیندهای :Received (بجز صندوق پستی فرستنده) باشد، ولی بندرت چنین است و معمولاً فقط آدرس فرستنده در آن نوشته می‌شود.

علاوه بر فیلدهای شکل ۷-۹، پیامهای RFC 822 دارای سرآیندهای دیگری نیز هستند که به بیشتر کار عامل کاربر یا شخص گیرنده می‌آیند. در شکل ۷-۱۰ برخی از این سرآیندها را می‌بینید. کارکرد اغلب این فیلدها از روی نامشان پیداست، و نیازی به توضیح زیاد ندارند.

سرآیند	مفهوم
Date:	زمان و تاریخ ارسال
Reply-To:	آدرس ایمیل برای پاسخ نامه
Message-Id:	عدد منحصر بفرد شناسایی پیام
In-Reply-To:	شماره پیامی که این پاسخ پیام پاسخ آن است
References:	سایر شماره های مربوطه
Keywords:	کلمات کلیدی انتخاب شده توسط کاربر
Subject:	موضوع پیام

شکل ۷-۱۰. برخی از فیلدهای سرآیند RFC 822 .

فیلد :Reply-To برای مواقعیست که نویسنده و گیرندهٔ نامه هیچکدام نمی‌خواهند گیرندهٔ پاسخ نامه باشند. برای مثال، وقتی مدیر بازاریابی نامه‌ای دربارهٔ محصولات جدید شرکت به یک مشتری می‌نویسد، و منشی هم آنرا می‌فرستد، فیلد :Reply-To می‌تواند به آدرس قسمت فروش شرکت، که پاسخگوی سفارشات هستند، اشاره کند. این فیلد برای مواردی که فرستنده دو آدرس ایمیل دارد، و مایل است پاسخها را از طریق آدرس دیگرش بگیرد، نیز مفید است.

استاندارد RFC 822 اجازه می‌دهد تا کاربران سرآیندهای دلخواهشان را به نامه‌ها اضافه کنند، مشروط باینکه این سرآیندها با -X شروع شوند (هیچیک از سرآیندهای رسمی این استاندارد با -X شروع نمی‌شوند، و در آینده نیز نخواهند شد). این قبیل سرآیندها می‌توانند اطلاعات اضافی را با خود حمل کنند.

بعد از سرآیند، بدنهٔ نامه می‌آید. کاربران می‌توانند هر چیزی در این بدنه بنویسند. برخی افراد در انتهای نامه‌هایشان اختتامیه‌های ماهرانه‌ای می‌آورند، مانند اشکال کارتونی جالب و خنده‌دار، کلمات قصار مشاهیر، و یا عبارات قانونی سلب مسئولیت (مثلاً، «شرکت فلان و بهمان هیچگونه مسئولیتی را دربارهٔ محتویات این نامه نمی‌پذیرد.»).

MIME - الحاقات چندمنظورهٔ پُست اینترنت

در روزهای اولیهٔ آرپانت، ایمیل‌ها فقط متن ساده بود که به زبان انگلیسی و با فرمت ASCII نوشته می‌شد. استاندارد RFC 822 با این وضعیت هیچ مشکلی نداشت: کاربر می‌توانست هر چیزی که می‌خواست در بدنهٔ نامه بنویسد. اما این روش دیگر برای دنیای امروز کافی نیست. برخی از مشکلات ذاتی این سیستم عبارتند از:

۱. ارسال پیام به زبانهایی که إعراب دارند (مانند فرانسه و آلمانی).
۲. ارسال پیام به زبانهای غیرلاتین (مانند عربی و روسی).
۳. ارسال پیام به زبانهای غیرالفبایی (مانند چینی و ژاپنی).
۴. ارسال پیامهایی که اصلاً متن نیستند (مانند صدا و تصویر).

راه حل این مشکلات در RFC 1341 ارئه شد، و بعدها در RFC 2045-49 به روز در آمد. این راه حل که MIME (الحاقات چندمنظورهٔ پُست اینترنت – Multipurpose Internet Mail Extensions) نام دارد، امروزه بطور گسترده‌ای رواج یافته است.

ایدهٔ اصلی MIME عبارتست از: ادامهٔ استفاده از فرمت RFC 822، و اضافه کردن ساختاری جدید به بدنهٔ پیام و تعریف قواعد درج پیامهای غیرمتنی. پیامهای MIME با برنامه‌ها و پروتکلهای موجود ایمیل کاملاً سازگارند، چون از استاندارد RFC 822 تخطی نمی‌کنند. فقط برنامه‌های عامل کاربر باید عوض شوند، که این کار را هم کاربران می‌توانند براحتی انجام دهند.

MIME پنج سرآیند جدید تعریف می‌کند، که آنها را در شکل ۷-۱۱ مشاهده می‌کنید. اولین سرآیند به عامل کاربر دریافت کنندهٔ پیام می‌گوید که با یک پیام MIME سروکار دارد، و ویرایش آن هم اعلام می‌شود. هر پیامی که سرآیند *MIME-Version:* نداشته باشد، متن ساده تلقی شده و به همان طریق پردازش می‌شود.

سرآیند	مفهوم
MIME-Version:	ویرایش MIME پیام
Content-Description:	جمله ای درباره محتویات پیام
Content-Id:	عدد منحصر بفرد شناسایی محتویات پیام
Content-Transfer-Encoding:	نحوه کدگذاری پیام
Content-Type:	نوع و فرمت محتویات پیام

شکل ۷-۱۱. سرآیندهای اضافی MIME در RFC 822 .

سرآیند *Content-Description:* یک عبارت متنی است که می‌گوید چه چیزی در پیام وجود دارد. از روی این سرآیند است که گیرنده تشخیص می‌دهد آیا محتویات پیام ارزش خواندن دارد یا خیر. برای مثال، اگر سرآیند *Content-Description:* بگوید: «این عکس یک موش است»، و گیرنده علاقه‌ای به دیدن عکس موش نداشته باشد، برنامه پیام را دور انداخته و تلاشی برای نمایش این عکس نخواهد کرد.

سرآیند *Content-Id:* محتویات پیام را مشخص می‌کند. فرمت این سرآیند مانند *Message-Id:* است.

سرآیند *Content-Transfer-Encoding:* نحوهٔ کُد شدن محتویات پیام (برای گذر از شبکه‌هایی که فقط به متن ساده اجازهٔ عبور می‌دهند) را مشخص می‌کند. پنج نوع کُدگذاری پیام وجود دارد. نوع اول فقط متن سادهٔ ASCII است. کاراکترهای ASCII هفت‌بیتی هستند، و تمام پروتکلهای ایمیل می‌توانند خطوط متن را (مشروط بر اینکه هر خط از ۱۰۰۰ حرف تجاوز نکند) منتقل کنند.

نوع دوم در واقع همان نوع قبلیست، که فقط از کاراکترهای ASCII ۸بیتی (از 0 تا 255) استفاده می‌کند.

برخی از بخشهای اینترنت از این کُد برای ارسال متن (و نمایش کاراکترهای خاص) استفاده می‌کنند. این کُدگذاری در پروتکلهای ایمیل اینترنتی مجاز نیست (و این تعریف هم باعث مجاز شدن آن نمی‌شود)، ولی حداقل توضیح می‌دهد که اشکال کار از کجاست. پیامهای دارای کُدگذاری ۸ بیتی هم محدود به خطهای ۱۰۰۰ حرفی هستند.

بدتر از آن پیامهایی هستند که کُدگذاری باینری دارند. اینها فایلهای باینری هستند، که نه تنها از تمام حالات ممکنهٔ ۸ بیت استفاده می‌کنند، بلکه به محدودیت ۱۰۰۰ بایت نیز وقعی نمی‌گذارند. برنامه‌های اجرایی در این دسته قرار می‌گیرند. هیچ تضمینی وجود ندارد که پیامهای باینری درست به مقصد برسند، ولی بسیاری افراد همچنان کار خودشان را می‌کنند.

یکی از روشهای کُدگذاری صحیح پیامهای باینری روش **کُدگذاری base64** (که گاهی ASCII armor نیز گفته می‌شود) است. در این روش، هر دستهٔ ۲۴ بیتی به چهار واحد ۶ بیتی تقسیم شده، و هر واحد بعنوان یک کاراکتر معتبر ASCII فرستاده می‌شود. در این روش "A" معادل 0 است، "B" معادل 1 ، ... و بالاخره "Z" معادل 26 ؛ پس از آن ۲۶ حرف کوچک انگلیسی می‌آیند، و پس از آن ارقام 0 تا 9 ؛ 62 و 63 نیز بترتیب معادل + و / هستند. توالیهای == و = بترتیب نشان می‌دهند که آخرین گروه ۸ یا ۱۶ بیتی است. کاراکترهای «برگشت سر خط» (carriage return) و «خط بعدی» (line feed) نیز بکلی نادیده گرفته می‌شود، بنابراین می‌توان برای کوتاه کردن خطوط از آنها استفاده کرد. با این روش می‌توان فایلهای باینری را بطور صحیح ارسال کرد.

برای پیامهایی که تقریباً بطور کامل ASCII هستند و فقط چند کاراکتر غیر ASCII دارند، کُدگذاری base64 کارایی مطلوبی ندارد. برای این قبیل پیامها از **کُدگذاری quoted-printable** استفاده می‌شود. این در واقع همان روش ASCII ۷ بیتی است، که در آن کاراکترهای بالای 127 با علامت = و یک عدد هگزادسیمال دو رقمی مشخص می‌شوند.

اطلاعات باینری همواره باید با یکی از روشهای base64 یا quoted-printable فرستاده شوند. اگر دلیل موجهی برای استفاده نکردن از هر یک از روشها وجود داشته باشد، می‌توان از کُدگذاریهای خاص (که با سرآیند *Content-Transfer-Encoding:* مشخص می‌شوند) استفاده کرد.

آخرین آیتم شکل ۷-۱۱ در واقع مهمترین سرآیند MIME است. این سرآیند خصلت واقعی محتویات پیام را مشخص می‌کند. در RFC 2045 هفت نوع (type) تعریف شده، که هر کدام از آنها می‌توانند چندین زیرنوع (subtype) داشته باشند. نوع و زیرنوع با یک / از هم جدا می‌شوند:

```
Content-Type: video/mpeg
```

زیرنوع باید صریحاً در سرآیند مشخص شود: هیچ مقداری بعنوان پیش‌فرض وجود ندارد. لیست اولیهٔ نوع‌ها و زیرنوع‌های RFC 2045 را در شکل ۷-۱۲ ملاحظه می‌کنید. از آن زمان تاکنون آیتمهای دیگری اضافه شده است، و در آینده باز هم اضافه خواهد شد.

اجازه دهید این لیست را مختصراً بررسی کنیم. نوع *text* همان متن ASCII ساده است. زیرنوع *text/plain* به گیرنده می‌گوید پیام را بهمان صورتی که دریافت کرده (بدون هیچ فرمت یا پردازشی) نمایش دهد. این گزینه اجازه می‌دهد تا پیامهای معمولی بدون هیچ اقدام اضافه‌ای به پیامهای MIME تبدیل شوند.

زیرنوع *text/enriched* اجازه می‌دهد تا از زبانهای علامتگذاری ساده برای فرمت کردن متن (تعیین فونت، اندازه، رنگ و صفحه‌بندی) استفاده شود. این زیرنوع زیرمجموعه‌ای از SGML (زبان علامتگذاری عام استاندارد - که زبان استاندارد وب یعنی HTML نیز جزیی از آن محسوب می‌شود) است. برای مثال پیام

```
The <bold> time </bold> has come the <italic> walrus </italic> said ...
```

به این صورت به نمایش در خواهد آمد:

نوع	زیرنوع	توضیح
Text	Plain	متن فرمت نشده
	Enriched	متن با فرمت ساده
Image	Gif	تصاویر با فرمت GIF
	Jpeg	تصاویر با فرمت JPEG
Audio	Basic	صدا
Video	Mpeg	تنظیم با فرمت MPEG
Application	Octet-stream	توالی بایت تفسیر نشده
	Postscript	سند چاپی با فرمت پست اسکریپت
Message	Rfc822	پیام RFC 822
	Partial	سند چند تکه شده
	External-body	پیام باید از اینترنت گرفته شود
Multipart	Mixed	پیامی با چند بخش مستقل
	Alternative	یک پیام با فرمتهای مختلف
	Parallel	بخشهای پیام باید همزمان دیده شوند
	Digest	هر بخش یک پیام RFC 822 کامل است

شکل ۷-۱۲. نوع‌ها و زیرنوع‌های MIME تعریف شده در RFC 2045 .

The time has come the *walrus* said ...

فرمت کردن پیام بطور کامل بر عهدهٔ سیستم گیرنده است، و نباید انتظار داشته باشید چنین پیامی در تمام سیستمها یکسان (و آنطوری که شما تصور می‌کنید) دیده شود. گاهی یک سیستم معنای کاری را که شما خواسته‌اید نمی‌فهمد، و بجای آن کار دیگری انجام می‌دهد.

بعد از رواج وب، زیرنوع جدیدی بنام *text/html* (در RFC 2854) اضافه شد، که اجازه می‌داد صفحات وب از طریق ایمیل‌های RFC 822 فرستاده شوند. در RFC 3023 نیز زیرنوع دیگری (*text/xml*) برای ارسال پیامهای XML تعریف شده است. در ادامهٔ این فصل با HTML و XML بیشتر آشنا خواهید شد.

نوع MIME بعدی *image* است، که برای ارسال تصاویر ثابت بکار می‌رود. امروزه فرمتهای بسیار متنوعی برای ذخیره و ارسال کردن تصاویر (با فشرده‌سازی یا بدون آن) وجود دارد، که از میان آنها فرمتهای GIF و JPEG بسیار معروفند و تقریباً تمام مرورگرهای اینترنت از آنها پشتیبانی می‌کنند. (البته بعدها فرمتهای دیگری نیز به این لیست اضافه شد.)

نوع‌های *audio* و *video* بترتیب برای ارسال صدا و تصاویر متحرک هستند. توجه داشته باشید که نوع *video* فقط برای ارسال اطلاعات تصویری است، و اگر فایل شما دارای تراک صوتی هم باشد، باید آنها را جداگانه بفرستید. اولین فرمت ویدئویی که قابلیت ارسال از طریق ایمیل را پیداکرد، فرمت MPEG (گروه تخصصی تصاویر متحرک - Moving Picture Expert Group) بود؛ بعدها فرمتهای دیگر نیز اضافه شد. علاوه بر زیرنوع *audio/basic* ، در RFC 3003 زیرنوع دیگری (*audio/mpeg*) برای ارسال فایلهای صوتی MP3 تعریف شده است.

هر نوع فایل باینری دیگری که در دستجات بالا قرار نگیرد (و سیستم ایمیل نداند آنرا چگونه پردازش کند)، در ذیل نوع *application* دسته‌بندی خواهد شد. زیرنوع *octet-stream* فقط استریمی از بایتهاست (و سیستم هیچگونه تفسیری روی آنها نخواهد کرد). عامل کاربر بعد از دریافت این استریم، تنها کاری که می‌تواند انجام دهد ذخیره کردن آن بصورت یک فایل است - پس باید نام فایل را از کاربر بگیرد. پردازشهای بعدی نیز بر عهدهٔ کاربر است.

زیرنوع تعریف شدهٔ دیگر *postscript* است، که به زبان پُست اسکریپت (زبان توصیف صفحات چاپی، از شرکت Adobe) مربوط می‌شود. بسیاری از چاپگرهای امروزی دارای مفسرهای پُست اسکریپت هستند. با اینکه عامل کاربر می‌تواند تفسیر فایلهای پُست اسکریپت را بر عهدهٔ برنامه‌های خارجی بگذارد، اما این کار خالی از خطر نیست. پُست اسکریپت یک زبان برنامه‌نویسی کامل است، و یک برنامه‌نویس (خودآزار) می‌تواند با صرف وقت کافی حتی یک کامپایلر C یا سیستم مدیریت پایگاه داده با آن بنویسد. عامل کاربر برای نمایش محتویات فایل پُست‌اسکریپت، در واقع برنامه‌ای راکه در دل پیام فرستاده شده اجرا می‌کند. چنین برنامه‌ای حتی می‌تواند (در کنار نمایش یک متن ساده) فایلهای کاربر را تغییر داده، یا آنها را پاک کند (و یا دهها کار ناجور دیگر انجام دهد).

نوع *message* اجازه می‌دهد تا یک پیام را بطور کامل در دل پیام دیگر جای دهیم. این نوع بویژه برای هدایت پیامها (message forwarding) مفید است. برای قرار دادن یک ایمیل RFC 822 در دل پیام دیگر، می‌توان از زیرنوع *message/rfc822* استفاده کرد.

با زیرنوع *partial* می‌توان یک پیام را چند تکه کرده، و هر تکه را در دل یک ایمیل جداگانه قرار داد (که این برای پیامهای خیلی بزرگ مناسب است). در مقصد سیستم ایمیل می‌تواند با استفاده از پارامترهای این زیرنوع، پیام تکه‌تکه شده را دوباره به هم بچسباند.

بالاخره، از زیرنوع *external-body* می‌توان برای پیامهای بسیار بزرگ (مانند فیلمهای ویدئویی) استفاده کرد؛ یعنی بجای قرار دادن فایل MPEG در دل پیام، آدرس FTP آن نامه در نوشته می‌شود، و عامل کاربر می‌تواند در موقع نیاز به این آدرس مراجعه کرده و فایل را بخواند. این زیرنوع بخصوص در مواردی که بخواهیم فایل بزرگی را برای یک لیست پستی بفرستیم، بسیار ایده‌آل است، چون احتمال اینکه همهٔ اعضاء لیست پستی این فایل را بخواهند چندان زیاد نیست (فرستادن آگهی‌های تبلیغاتی یکی از این موارد است).

نوع آخر *multipart* است، که اجازه می‌دهد یک پیام چندین بخش داشته باشد (بخشهایی که ابتدای و انتهای آنها کاملاً مشخص است). در زیرنوع *mixed* این بخشها می‌توانند کاملاً متفاوت باشند، بدون اینکه نیازی به ساختار اضافی وجود داشته باشد. در بسیاری از برنامه‌های ایمیل یک پیام ساده می‌تواند چندین پیوست (attachment) از انواع مختلف داشته باشد، که این پیوستها با استفاده از نوع *multipart* فرستاده می‌شوند.

بر خلاف *multipart* ، در زیرنوع *alternative* یک پیام واحد به چندین فرمت (مانند متن ساده، متن فرمت‌دار، و یا پست اسکریپت) فرستاده می‌شود، که عامل کاربر گیرنده می‌تواند بسته به امکانات خود از یکی از این انواع استفاده کند. این انواع باید از ساده‌ترین به پیچیده‌ترین مرتب شوند، تا حتی کاربران غیر MIME بتوانند پیامها را بخوانند.

از زیرنوع *alternative* برای ارسال پیام به زبانهای مختلف هم می‌توان استفاده کرد. (سنگ روزِتا را می‌توان یکی از قدیمیترین پیامهای *multipart/alternative* دانست - سنگ روزِتا سنگ‌نبشته‌ای است که توسط سپاهیان ناپلئون در منطقه‌ای به همین نام در مصر کشف شد، و در آن یک پیام واحد به زبانهای هیروگلیف و یونانی نوشته شده بود؛ با استفاده از همین سنگ‌نبشته بود که شامپولیون باستان‌شناس فرانسوی توانست معمای خط هیروگلیف را بعد از قرنها کشف کند.)

در شکل ۷-۱۳ یک پیام *multipart* را ملاحظه می‌کنید. در این مثال، یک پیام تبریک تولد به دو صورت متن و صوت فرستاده شده است. اگر کامپیوتر گیرنده کارت صوتی داشته باشد، برنامهٔ عامل کاربر فایل صوتی *birthday.snd* را از شبکه گرفته، و پخش می‌کند. اما اگر چنین نباشد، فقط متن شعر را نمایش خواهد داد. دقت کنید که بخشهای پیام با دو - (خط تیره) و رشته‌ای از حروف (که نرم‌افزار آنها را تولید کرده، و در قسمت *boundary* مشخص شده) از هم جدا می‌شوند.

همچنین توجه کنید که سرآیند *Content-Type* در سه نقطه از این پیام آمده است. در بالای پیام، سرآیند

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched

Happy birthday to you
Happy birthday to you
Happy birthday dear <bold> Carolyn </bold>
Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.abcd.com";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

شکل ۷–۱۳. یک پیام *multipart* ، محتوی متن و صوت.

Content-Type می‌گوید که این یک پیام *multipart/alternative* است. در دو *Content-Type* بعدی نوع و زیرنوع هر بخش مشخص می‌شود. در آخرین *Content-Type* نیز کُدگذاری فایل صوتی را مشخص کرده‌ایم، چون هر چیزی که متن ASCII ۷بیتی نباشد، باید دارای کُدگذاری مشخص باشد.

پیامهای *multipart* دارای دو زیرنوع دیگر نیز می‌توانند باشند. از زیرنوع *parallel* وقتی استفاده می‌کنیم که بخواهیم تمام بخشهای پیام همزمان «مشاهده» شوند. برای مثال، فیلمهای ویدئویی دارای کانالهای تصویری و صوتی مجزا هستند، که باید همزمان پخش شوند (نه بدنبال هم).

زیرنوع *digest* وقتی بکار برده می‌شود که بخواهیم تعدادی پیام را در یک پیام مرکب بسته‌بندی کنیم. برای مثال، در **گروههای مباحثه** (discussion group) اینترنتی معمولاً چندین پیام که از اعضای مختلف جمع‌آوری شده، در یک پیام *multipart/digest* به سایر اعضای گروه فرستاده می‌شود.

۷-۲-۴ انتقال پیام

سیستم انتقال پیام (message transfer) با ارسال پیام از فرستنده به گیرنده سروکار دارد. ساده‌ترین راه برای این کار، برقراری یک اتصال مستقیم از ماشین فرستنده به ماشین گیرنده، و انتقال پیام است. بعد از بررسی این روش، به مواردی می‌پردازیم که چنین کاری امکان ندارد، و سپس برای آن موارد نیز راه‌حل‌هایی نشان خواهیم داد.

SMTP - پروتکل سادهٔ انتقال نامه

در اینترنت، انتقال ایمیل با برقراری یک اتصال TCP از ماشین مبدأ به پورت 25 ماشین مقصد صورت می‌گیرد. برنامه‌ای که به این پورت گوش می‌کند، دیمون SMTP (پروتکل سادهٔ انتقال نامه - Simple Mail Transfer Protocol) نام دارد. این دیمون اتصالات ورودی را پذیرفته، و پیامها را در صندوق پستی مربوطه کپی می‌کند. اگر گیرنده نتواند پیامی را تحویل گیرد، یک گزارش خطا حاوی اولین بخش از پیام مزبور به فرستنده برمی‌گرداند.

SMTP یک پروتکل سادهٔ ASCII است. بعد از برقراری اتصال TCP ، ماشین فرستنده (که نقش مشتری را بازی می‌کند) منتظر می‌ماند تا ماشین گیرنده (که نقش سرویس‌دهنده را بازی خواهد کرد) شروع به صحبت کند. در شروع، سرویس‌دهنده یک خط متن فرستاده و ضمن معرفی خود، اعلام می‌کند که آیا آماده دریافت ایمیل هست یا خیر. اگر سرویس‌دهنده آماده نباشد، مشتری ارتباط را قطع کرده، و بعداً دوباره سعی خواهد کرد.

اگر سرویس‌دهنده آمادهٔ دریافت باشد، مشتری اعلام می‌کند که پیام از چه کسی می‌آید و به چه کسی باید تحویل شود. اگر چنین دریافت‌کننده‌ای در ماشین سرویس‌دهنده وجود داشته باشد، سرویس‌دهنده از مشتری می‌خواهد که پیام را بفرستد. پس از آن که مشتری پیام را فرستاد، سرویس‌دهنده دریافت آن را تصدیق می‌کند. سرویس‌دهنده هیچ تلاشی برای چک کردن جمع تطبیقی انجام نخواهد داد، چون TCP سالم بودن ارتباط را تضمین می‌کند. اگر باز هم پیامی وجود داشته باشد، پس از آن فرستاده می‌شود. وقتی تمام ایمیل‌ها (در هر دو جهت) مبادله شد، ارتباط قطع می‌شود. در شکل ۷-۱۴ مکالمهٔ سرویس‌دهنده و مشتری (منجمله کُدهای عددی SMTP) برای ارسال ایمیل شکل ۷-۱۳ را ملاحظه می‌کنید. خطوطی که توسط مشتری ارسال شده‌اند، را با :C ، و آنهایی که توسط سرویس‌دهنده فرستاده شده‌اند، را با :S مشخص کرده‌ایم.

کمی توضیح دربارهٔ شکل ۷-۱۴ می‌تواند مفید باشد. اولین پیام مشتری *HELO* است. این کلمه در واقع مخفف چهارحرفی «سلام» *(HELLO)* است، و پیداست که از همهٔ چهارحرفی‌ها به آن شبیه‌تر است. اینکه چرا باید از کلمه‌های چهارحرفی استفاده کنیم، در غبار زمان گم شده، ولی این رسم همچنان پا برجاست.

از آنجائیکه فقط یک گیرنده وجود دارد، در اینجا فقط یک دستور *RCPT* دیده می‌شود، ولی می‌توان در آن واحد یک پیام را به چندین گیرنده فرستاد (که قبول یا رد درخواست برای هر یک جداگانه انجام خواهد شد).

با اینکه دستورات چهارحرفی از طرف مشتری ثابت و مشخص هستند، پاسخ سرویس‌دهنده فقط یک سری کُدهای عددی است. در واقع همین کُدهاست که اهمیت دارد، و هر سیستم می‌تواند برای هر کُد توضیح خاص خود را داشته باشد.

اجازه دهید برای درک بهتر پروتکل SMTP ، کمی دربارهٔ فرآیند کار توضیح دهیم. قبل از هر چیز سراغ کامپیوتری بروید که به اینترنت دسترسی دارد. سپس دستور زیر را در خط فرمان وارد کنید (در این مثال فرض کرده‌ایم سیستم عامل یونیکس است)

```
telnet mail.isp.com 25
```

(بجای mail.isp.com آدرس IP یا نام DNS سیستم ایمیل خود را وارد کنید.) در سیستمهای ویندوز، پنجرهٔ Start|Run را باز کرده و دستور فوق را در آنجا وارد کنید. این دستور یک اتصال telnet (یعنی، TCP) با پورت 25 ماشین مشخص شده برقرار می‌کند. پورت 25 پورت SMTP است (شکل ۶-۲۷ را ببینید). پاسخی که دریافت خواهید کرد، احتمالاً چیزی شبیه زیر است:

```
Trying 192.30.200.66...
Connection to mail.isp.com
Escape character is '^]'
220 mail.isp.com Smail #74 ready at Thu, 25 Mar 2003 13:26 +0200
```

```
                    S: 220 xyz.com SMTP service ready
C: HELO abcd.com
                    S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
                    S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
                    S: 250 recipient ok
C: DATA
                    S: 354 Send mail; end with "." on a line by itself
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Carolyn </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:         access-type="anon-ftp";
C:         site="bicycle.abcd.com";
C:         directory="pub";
C:         name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                    S: 250 message accepted
C: QUIT
                    S: 221 xyz.com closing connection
```

شکل ۷-۱۴. انتقال یک پیام از *elinor@abcd.com* به *carolyn@xyz.com*.

سه خط اول مربوط به برنامهٔ telnet هستند، و می‌گویند چه اتفاقی در حال افتادن است. خط آخر از سرویس‌دهنده SMTP آمده، و آمادگی آنرا برای دریافت ایمیل از طرف شما اعلام می‌کند. برای اینکه ببینید سرویس‌دهنده ایمیل چه فرمانهایی را قبول می‌کند، دستور زیر را وارد کنید

HELP

از اینجا به بعد با چیزی شبیه شکل ۷-۱۴ روبرو خواهید بود. پروتکلهای ASCII در اینترنت بسیار رایج هستند، چون تست و دیباگ آنها بسیار ساده است. فرمان دادن به این پروتکلها بسیار آسان است، و پاسخ آنها را نیز براحتی می‌توان درک کرد.

با اینکه پروتکل SMTP بخوبی تعریف شده است، اما برخی مشکلات کوچک نیز می‌تواند بروز کند. یکی از این مشکلات طول پیام است: در برخی از سیستمهای ایمیل قدیمی طول پیام نباید از 64 KB تجاوز کند. مشکل دیگر زمانهای متفاوت انتظار برای پاسخ در دو سمت مقابل است. اگر زمان انتظار برای دریافت پاسخ (timeout) در سمت سرویس‌دهنده و مشتری متفاوت باشد، این احتمال هست که یکی از آنها تسلیم شده (در حالیکه دیگری هنوز منتظر است) و ارتباط را بطور نامنتظره قطع کند. مشکل دیگر بروز توفانهای ایمیل است. اگر ماشین ۱ دارای یک لیست پستی بنام A ، و ماشین ۲ دارای یک لیست پستی بنام B باشند، و هر لیست عضو لیست مقابل باشد، توفانی پایان‌ناپذیر از ایمیلهای تکراری به راه خواهد افتاد (که فقط با دخالت سرپرست سیستم می‌تواند قطع شود).

برای حل این قبیل مشکلات، ویرایش گسترش‌یافتهٔ SMTP (موسوم به ESMTP) در RFC 2821 تعریف شده است. اگر یک مشتری بخواهد بجای SMTP از این پروتکل استفاده کند، باید در شروع کار بجای *HELO* دستور *EHLO* را بکار ببرد. اگر دستور *EHLO* از طرف سرویس‌دهنده پذیرفته نشود، مشتری می‌فهمد که با یک سرویس‌دهنده *SMTP* معمولی سروکار دارد و باید از همان روش سابق استفاده کند. اما اگر *EHLO* پذیرفته شد، مشتری اجازه دارد دستورات این پروتکل را بکار ببرد.

۷-۲-۵ تحویل نهایی

تا اینجا فرض ما بر این بود که تمام کاربران شبکه ماشینهایی با قابلیت ارسال و دریافت ایمیل دارند. همانطور که دیدید، فرستنده باید یک اتصال TCP به ماشین گیرنده برقرار کرده، و ایمیل خود را به آن بفرستد. این روش سالها بخوبی کار می‌کرد، چون تمام ماشینهای آرپانت (و بعدها اینترنت) کامپیوترهایی بودند که همیشه روی خط بودند و می‌توانستند اتصالات TCP را بپذیرند.

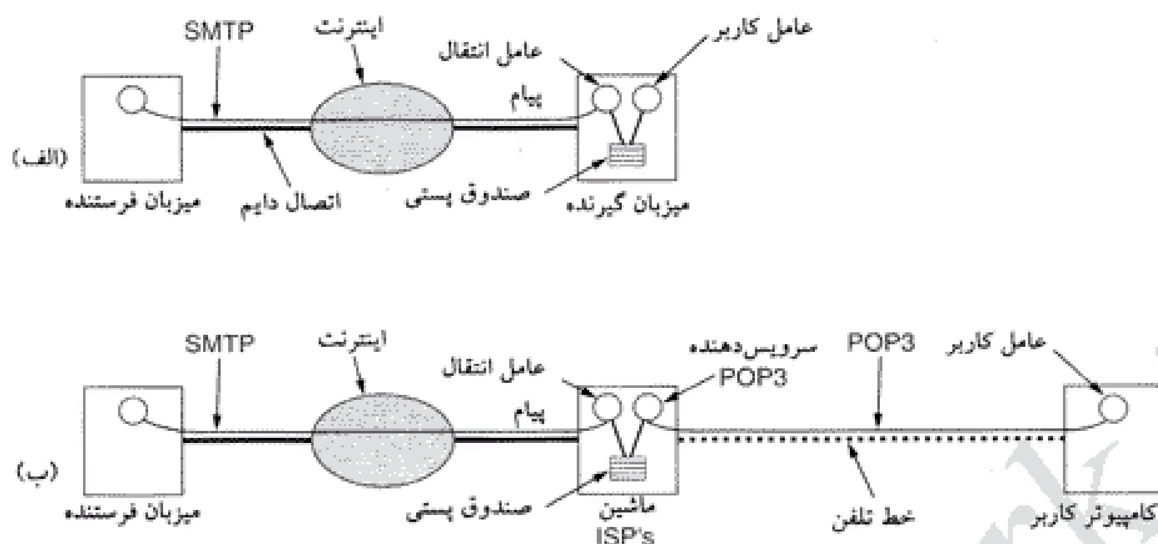
این وضعیت با ظهور کاربرانی که برای دسترسی اینترنت مجبور بودند از طریق مودم و با واسطهٔ یک ISP اقدام کنند، تغییر کرد. مشکل این بود که: اگر در لحظه‌ای که Elinor می‌خواهد برای Carolyn ایمیل بفرستد، Carolyn روی خط نباشد، چه خواهد شد؟ در این حالت Elinor قادر به برقراری ارتباط TCP با Carolyn نیست، و نمی‌تواند پروتکل SMTP را اجرا کند.

ساده‌ترین راه‌حل آن است که یک عامل انتقال پیام در محل ISP ایجاد شود، و ایمیلها را پس از دریافت از فرستنده در صندوق پستی گیرنده (که در همان ISP قرار دارد) ذخیره کند. از آنجائیکه این عامل انتقال پیام می‌تواند ۲۴ ساعته روی خط باشد، همیشه می‌توان به آن ایمیل فرستاد.

POP3

متأسفانه این راه‌حل یک مشکل کوچک دارد: کاربران چگونه باید ایمیل‌های خود را از عامل انتقال مستقر در ISP بگیرند؟ برای حل این مسئله به پروتکل جدیدی نیاز داریم که PC کاربر را به یک عامل انتقال پیام (از ISP) تبدیل کند. یکی از این پروتکل‌ها، که در RFC 1939 تعریف شده، POP3 (**پروتکل دفترپستی ویرایش ۳** – Post Office Protocol Version 3) نام دارد.

به شکل ۷-۱۵ نگاه کنید؛ در تصویر (الف) وضعیتی که باید وجود داشته باشد، را می‌بینید: فرستنده و گیرنده



شکل ۷-۱۵. (الف) وضعیتی که فرستنده و گیرنده دسترسی دائم به اینترنت دارند، و عامل کاربر و عامل انتقال پیام هر دو روی یک ماشین اجرا می‌شوند. (ب) خواندن ایمیل در حالتی که گیرنده از طریق تلفن و با واسطهٔ یک ISP به اینترنت وصل می‌شود.

هر دو دائماً روی خط هستند. در شکل ۷-۱۵ (ب) اوضاع به این خوبی نیست: فرستنده همیشه روی خط است، ولی گیرنده چنین نیست.

پروتکل POP3 کار خود را زمانی شروع می‌کند که کاربر برنامهٔ ایمیل‌خوان را باز می‌کند. برنامهٔ ایمیل‌خوان با ISP تماس گرفته (البته اگر این تماس از قبل وجود نداشته باشد)، و یک اتصال TCP به پورت 110 ماشین عامل انتقال پیام برقرار می‌کند. بعد از برقراری ارتباط، پروتکل POP3 سه مرحله را طی می‌کند:

۱. احراز هویت (authorization) .
۲. تراکنش (transaction) .
۳. به روز در آوردن (update) .

در مرحلهٔ احراز هویت کاربر باید هویت واقعی خود را به عامل انتقال پیام بشناساند. در مرحلهٔ تراکنش، کاربر ایمیل‌های خود را از صندوق پستی خوانده، و آنها را برای حذف شدن علامتگذاری می‌کند. این ایمیل‌ها در مرحلهٔ بعد (به روز در آوردن) حذف می‌شوند.

برای دیدن مراحل کار، فرمان زیر را اجرا کنید:

```
telnet mail.isp.com 110
```

(که در آن *mail.isp.com* نام DNS سرویس‌دهنده ایمیل ISP است.) برنامهٔ telnet یک اتصال TCP با پورت 110 (که سرویس‌دهنده POP3 به آن گوش می‌کند) برقرار می‌سازد. بعد از برقراری ارتباط، سرویس‌دهنده یک پیام ASCII فرستاده، و آمادگی خود را اعلام می‌کند. این پیام معمولاً با OK+ شروع می‌شود، و جملهٔ کوتاه دیگری بعد از آن می‌آید. در شکل ۷-۱۶ مکالمهٔ یک مشتری با سرویس‌دهنده POP3 را ملاحظه می‌کنید (در اینجا هم پیامهای مشتری را با :C و پاسخهای سرویس‌دهنده را با :S مشخص کرده‌ایم).

در مرحلهٔ احراز هویت، مشتری ابتدا **نام‌کاربر** (username) و سپس **کلمهٔ‌عبور** (password) خود را می‌فرستد. اگر مشتری بتواند این مرحله را با موفقیت پشت سر گذارد، می‌تواند با ارسال دستور *LIST* فهرستی از پیامهای موجود در صندوق پستی خود را دریافت کند (یک پیام در هر خط، که طول آنها نیز مشخص شده است).

پایان این لیست با یک نقطه (.) مشخص می‌شود.

```
                S: +OK POP3 server ready
C: USER carolyn
                S: +OK
C: PASS vegetables
                S: +OK login successful
C: LIST
                S: 1 2505
                S: 2 14302
                S: 3 8122
                S: .
C: RETR 1
                S: (sends message 1)
C: DELE 1
C: RETR 2
                S: (sends message 2)
C: DELE 2
C: RETR 3
                S: (sends message 3)
C: DELE 3
C: QUIT
                S: +OK POP3 server disconnecting
```

شکل ۷-۱۶. استفاده از POP3 برای گرفتن پیامهای ایمیل.

پس از آن مشتری می‌تواند برای دریافت پیامها از فرمان *RETR* استفاده کرده، و آنها را با *DELE* برای حذف علامتگذاری کند. وقتی تمام پیامها دریافت (و احتمالاً برای حذف علامتگذاری) شدند، مشتری می‌تواند با فرمان *QUIT* وارد مرحلهٔ بعدی (به روز در آوردن) شود. بعد از آن که سرویس‌دهنده پیامهای علامتگذاری شده را حذف کرد، یک پیام تصدیق به مشتری فرستاده و ارتباط TCP را قطع می‌کند.

با اینکه پروتکل POP3 می‌تواند ایمیل‌ها را بصورت دسته‌جمعی یا تکی گرفته و آنها را روی سرویس‌دهنده‌باقی بگذارد، اغلب برنامه‌های ایمیل همه چیز را از روی سرویس‌دهنده خوانده و سپس صندوق پستی را خالی می‌کنند. در این حالت تنها کپی پیامها در دست خود کاربر است، و اگر روزی کامپیوتر وی صدمه ببیند، همهٔ ایمیل‌ها از بین خواهند رفت.

اجازه دهید یک بار دیگر روش ارسال و دریافت ایمیل را برای کاربران ISP بطور خلاصه مرور کنیم. Elinor در برنامهٔ ایمیل خود (یعنی، عامل کاربر) یک نامه برای Carolyn می‌نویسد، و روی دکمهٔ Send کلیک می‌کند. برنامهٔ ایمیل این پیام را گرفته و به عامل انتقال پیام ISP (که Elinor مشتری آن است) تحویل می‌دهد. عامل انتقال پیام آدرس گیرندهٔ نامه (*carolyn@xyz.com*) را خوانده، و از یک DNS کمک می‌گیرد تا رکورد *MX* ناحیهٔ *xyz.com* را پیدا کند. درپاسخ به این درخواست، DNS نام DNS سرویس‌دهنده ایمیل ناحیهٔ *xyz.com* را برمی‌گرداند. عامل انتقال پیام دوباره به کمک همان DNS آدرس IP این سرویس‌دهنده را درخواست می‌کند. پس از دریافت این آدرس IP ، عامل انتقال پیام یک اتصال TCP به پورت 25 (سرویس‌دهنده SMTP) برقرار می‌کند، و با استفاده از دستورات SMTP (مانند آنچه در شکل ۷-۱۴ دیدید) پیام را به صندوق پستی Carolyn فرستاده، و سپس ارتباط TCP را قطع می‌کند.

پس از مدتی، Carolyn (از خواب بیدار شده و) PC خود را روشن کرده، و مثل همیشه قبل از هر کاری به ISP وصل می‌شود تا ایمیل‌هایش را چک کند. برنامهٔ ایمیل در بدو شروع یک اتصال TCP به پورت 110 سرویس‌دهنده ایمیل ISP (سرویس‌دهنده POP3) برقرار می‌کند. نام DNS یا آدرس IP این ماشین در همان بدو عضویت Carolyn در ISP در اختیار وی قرار داده می‌شود، تا آنرا در برنامهٔ ایمیل خود وارد کند (و معمولاً نیازی به پا در میانی DNS نیست). پس از برقراری اتصال به پورت 110 ، برنامهٔ ایمیل پروتکل POP3 را اجرا کرده، و (مانند آنچه در شکل ۷-۱۶ دیدید) پیامهای رسیده را از صندوق پستی Carolyn خوانده و در کامپیوتر وی ذخیره می‌کند (در پایان کار هم، اتصال TCP را قطع می‌کند). در اینجا می‌توان ارتباط تلفنی با ISP را هم قطع کرد (که البته نباید در این کار عجله کنید، چون برای فرستادن پاسخ نامه‌ها باز هم به آن احتیاج پیدا خواهد کرد).

IMAP

برای کاربری که فقط با یک ISP کار می‌کند و همیشه هم از یک PC به آن وصل می‌شود، POP3 بهترین گزینه است (بخصوص که ساده و کارآمد هم هست). ولی در دنیای کامپیوتر اصلی بدیهیست که می‌گوید، «وقتی چیزی دارد خوب کار می‌کند، بلافاصله یکی پیدا می‌شود که بیشتر می‌خواهد.» برای ایمیل هم همین اتفاق افتاد. برای مثال، خیلی از مردم هستند که فقط یک آدرس ایمیل دارند، و می‌خواهند هر کجا که هستند (در خانه، در مدرسه، در محل کار، و یا در سفر) با همان آدرس کار کنند. با اینکه POP3 می‌تواند از عهدهٔ این کار بر آید، اما کاربر بزودی متوجه می‌شود که ایمیل‌هایش روی چندین کامپیوتر پراکنده شده است (کامپیوترهایی که شاید بعضی از آنها حتی متعلق به وی نباشد).

این مشکل POP3 منجر به ارائهٔ راه‌حلی بنام IMAP (**پروتکل دسترسی پیام اینترنتی** - Internet Message Access Protocol) شد، که در RFC 2060 تعریف شده است. بر خلاف POP3 که اساساً فرض می‌کند کاربر تمام پیامهایش را به کامپیوتر خود منتقل کرده و سپس ارتباط با اینترنت را قطع می‌کند، IMAP پیامها را برای همیشه روی کامپیوتر سرویس‌دهنده نگه داشته و آنها را در چندین صندوق پستی حفظ می‌کند. IMAP دارای مکانیزمهای پیشرفته‌ای برای خواندن پیامهاست، که حتی اجازه می‌دهند کاربر فقط بخشهایی از یک پیام بخواند؛ فقط تصور کنید که یک پیام مهم برای کاربر بیچارهٔ ما - که با یک مودم کُند عهد بوق به اینترنت وصل می‌شود - آمده، و این پیام ضمن داشتن متنی مهم دارای یک پیوست بزرگ نیز هست - IMAP به کاربر ما اجازه می‌دهد تا فقط متن پیام را خوانده و از خیر پیوست آن بگذرد. از آنجائیکه در IMAP فرض بر آن است که پیامها به کامپیوتر کاربر منتقل نمی‌شوند، مکانیزمهایی برای نوشتن ایمیل، از بین بردن آنها، و یا مدیریت پیامهای رسیده (مانند دسته‌بندی آنها بر حسب فرستنده) روی سرویس‌دهنده ایمیل در نظر گرفته شده است.

یکی از قابلیتهای جالب IMAP دسته‌بندی و نمایش پیامهای رسیده بر حسب فرستندهٔ ایمیل - یا ویژگیهای دیگر - است (بر خلاف شکل ۷-۸ که تمام ایمیل‌ها پشت سر هم ردیف می‌شوند). IMAP (برخلاف POP3) فقط پروتکلی برای دریافت ایمیل نیست، بلکه می‌تواند ارسال نامه‌ها را هم انجام دهد. روش کار IMAP بسیار شبیه POP3 (شکل ۷-۱۶) است، با این تفاوت که دستورات بسیار متنوعتری دارد. سرویس‌دهنده IMAP به پورت 143 گوش می‌کند. در شکل ۷-۱۷ مقایسه‌ای بین POP3 و IMAP آورده شده است. اما همین جا باید تذکر داد که تمام ISP ها (و همچنین برنامه‌های ایمیل) از هر دو پروتکل پشتیبانی نمی‌کنند (هنگام انتخاب ISP و برنامهٔ ایمیل دقت کنید که از چه پروتکلهایی پشتیبانی می‌کنند).

امکانات سیستم تحویل نامه

اغلب سیستمهای ایمیل، صرفنظر از اینکه از POP3 یا IMAP برای گرفتن نامه‌ها استفاده کنید، امکاناتی برای

پردازش پیامها دارند. یکی از این امکانات فیلتر کردن پیامهاست. این فیلترها قواعدی هستند که روی ایمیلهای رسیده عمل می‌کنند. هر قاعده یک شرط دارد و عملی را انجام می‌دهد. برای مثال، یک قاعده می‌تواند بگوید «اگر پیامی از رئیس رسید، آنرا در صندوق شماره ۱ قرار بده.» یا «اگر پیام از گروهی مشخصی از دوستان بود، آنرا در صندوق شماره ۲ قرار بده.» و یا «اگر کلمهٔ خاصی در موضوع پیام بود، آنرا بکلی دور بینداز.»

ویژگی	POP3	IMAP
سطح تعریف پروتکل	RFC 1939	RFC 2060
پورت	110	143
محل ذخیره شدن ایمیل	کاربر PC	سرویس دهنده
محل خوانده شدن ایمیل	خارج خط	روی خط
زمان اتصال	کم	زیاد
استفاده از منابع سرویس دهنده	حداقل	گسترده
صندوق پستی های متعدد	خیر	بلی
مسئول گرفتن پشتیبان	کاربر	ISP
مناسب برای کاربران سیار	خیر	بلی
کنترل بار کردن محتویات	کم	زیاد
بار کردن قسمتی از پیامها	خیر	بلی
مشکل محدودیت دیسک	خیر	گاهی
پیاده سازی ساده است	بلی	خیر
پشتیبانی گسترده	بلی	در حال رشد

شکل ۷-۱۷. مقایسه‌ای بین POP3 و IMAP .

برخی از ISP ها فیلترهایی دارند که بطور خودکار ایمیلهای رسیده را به دو دستهٔ «مهم» و «هرز» (spam یا junk – نامه‌هایی که فقط بدرد سطل آشغال می‌خورند) تقسیم کرده، و آنها را در صندوقهای مخصوص قرار می‌دهند. این فیلترها ابتدا با چک کردن فرستندهٔ نامه‌ها شروع می‌کنند (تا مطمئن شوند از مردم‌آزارهای حرفه‌ای نباشند). اگر چند صد کاربر یک ISP نامه‌هایی با یک موضوع دریافت کنند، فرستندهٔ آن بااحتمال زیاد یک هرزنویس است. برای تشخیص این قبیل نامه‌ها تکنیکهای دیگری نیز وجود دارد.

یک دیگر از امکانات سیستمهای تحویل ایمیل هدایت (موقتی) نامه‌های رسیده برای یک کاربر به آدرسی دیگر است. این آدرس حتی می‌تواند شماره‌ای در یک سیستم **فراخوان** (pager) باشد، که در این حالت کاربر بلافاصله بعد از دریافت هر ایمیل موضوع آنرا در دستگاه فراخوان خود مشاهده خواهد کرد.

امکان دیگر پاسخ خودکار در صورت عدم حضور در محل است (که به آن **دیمون تعطیلات** – vacation daemon – می‌گویند). با این ویژگی فرد می‌تواند هنگام رفتن به تعطیلات یا مأموریت بطور خودکار به ایمیلهای رسیده پاسخی مانند زیر بدهد:

سلام، من تا ۲۴ ام آگوست در تعطیلات تابستانی هستم. امیدوارم شما هم تعطیلات خوبی داشته باشد.

در این قبیل پیامها حتی می‌توانید مشخص کنید که در صورت اضطرار با کجا باید تماس گرفته شود. در برخی از سیستمهای ایمیل، دیمون تعطیلات می‌تواند تشخیص دهد که قبلاً برای چه کسانی پیام فرستاده، تا از ارسال پیام تکراری برای آنها اجتناب شود. برخی از این دیمون‌ها حتی می‌توانند تشخیص دهند که نامهٔ وارده به یک لیست پستی فرستاده شده، که در این صورت از فرستادن جواب آماده خودداری خواهند کرد.

نویسندهٔ این کتاب شخصاً با یک سیستم پاسخ خودکار فوق‌العاده جالب از جانب شخصی روبرو شده است،

که ادعا می‌کرد روزی ۶۰۰ ایمیل می‌گیرد. (اجازه دهید برای حفظ حریم شخصی افراد، این فرد را با نام مستعار جان معرفی کنیم.)

جان یک رویات ایمیل در کامپیوتر خود نصب کرده که تمام پیامهای رسیده را چک می‌کند. اگر این ایمیل از کسی باشد که قبلاً با جان تماس نداشته (و بعبارت دیگر تازه‌وارد باشد)، پیام خودکاری برای وی ارسال می‌شود که (ضمن عذرخواهی از عدم امکان پاسخ فردی) سندی حاوی تمام اطلاعات موردنیاز (از جمله آدرس، شماره تلفن، شماره فکس، و طریقهٔ تماس با شرکت جان) در آن آمده است. از اطلاعات مفصل دیگری که جان دربارهٔ خود در این پاسخ خودکار آورده حرفی نمی‌زنم، اما بنظر می‌رسد که روش جان یکی از نمونه‌های افراط در استفاده از امکانات باشد.

پُست وب

آخرین مبحثی که در سیستمهای ایمیل می‌توان به آن اشاره کرد، پُست‌وب (Webmail) است. همانطور که شاید قبلاً دیده باشید، برخی از سایتهای معروف و بزرگ مانند هات‌میل (Hotmail.com) و یاهو (Yahoo.com) سرویسهای ایمیل مجانی در اختیار بازدیدکنندگان قرار می‌دهند. در این سیستمها یک عامل انتقال پیام معمولی وجود دارد، که به پورت 25 (پورت SMTP) گوش می‌کند. برای وصل شدن به، مثلاً، هات‌میل باید ابتدا رکورد *MX* آنرا بدست آورید؛ در یک سیستم یونیکس می‌توانید از دستور زیر استفاده کنید:

```
host -a -v hotmail.com
```

پس از آن، با فرض اینکه سرویس‌دهنده ایمیل هات‌میل *mx10.hotmail.com* نام داشته باشد، می‌توانید با دستور

```
telnet mx10.hotmail.com 25
```

یک اتصال TCP به پورت 25 آن برقرار کرده، و از فرمانهای SMTP برای ارسال پیامهای خود استفاده کنید. (تا اینجا که چیز غیرعادی وجود ندارد؛ فقط مواظب باشید که سر این کامپیوترها خیلی شلوغ است، و معمولاً باید چند بار سعی کنید تا بتوانید با آنها تماس بگیرید.)

بخش جالب در این سرویسها قسمت تحویل نامه است. معمولاً وقتی وارد صفحهٔ وب سرویس ایمیل می‌شوید، فرمی ظاهر می‌شود که باید نام‌کاربر و کلمهٔ عبور خود را در آن وارد کنید. وقتی دکمهٔ Sign In را کلیک می‌کنید، این اطلاعات به سرویس‌دهنده فرستاده شده و در آنجا با اطلاعات موجود تطبیق داده می‌شود. اگر هویت شما تأئید شود، سرویس‌دهنده صندوق پستی را یافته و محتویات آنرا بصورت یک صفحه‌ای شبیه شکل ۷-۸، ولی با فرمت HTML ، نمایش می‌دهد. اغلب امکانات یک برنامهٔ ایمیل (مانند خواندن نامه، نوشتن نامه، و حذف آنها) در این صفحهٔ وب نیز وجود دارد.

۷-۳ تارنمای جهانی – وب

تارنمای جهانی (World Wide Web) ، یا بطور مختصر وب، ساختاریست برای دسترسی به **سندهای پیوندشده** (linked documents) در اینترنت. ظرف ده سال، وب از یک ابزار ارتباطی فیزیکدانها به چیزی تبدیل شده که بسیاری از مردم آنرا همان «اینترنت» می‌دانند. علت اصلی محبوبیت وب در ظاهر گرافیکی آن ریشه دارد، که باعث شده تا میلیونها کاربر تازه‌کار بتوانند بسادگی از آن استفاده کنند. حجم فوق‌العادهٔ اطلاعات موجود در وب را نیز می‌توان یکی دیگر از علل موفقیت آن دانست.

وب (که به WWW نیز معروف است) به سال ۱۹۸۹ در مرکز اروپایی فیزیک هسته‌ای موسوم به CERN متولد شد. در این مرکز دهها تیم تحقیقاتی از سراسر اروپا به کار روی فرضیه‌های فیزیک ذرات مشغول هستند.

اغلب این آزمایشات چنان پیچیده‌اند که دهها دانشمند از چندین کشور مختلف سالها روی آنها کار می‌کنند. همین پراکندگی دانشمندان این مرکز در کشورهای مختلف، و لزوم ارتباط پیوستهٔ آنها منجر به تولد وب شد.

ایدهٔ سندهای پیوند شده را اولین بار یکی از فیزیکدانان CERN بنام تیم برنرز-لی در مارس ۱۹۸۹ مطرح کرد. اولین نمونهٔ این برنامه ۱۸ ماه بعد عملیاتی شد، و در کنفرانس Hypertext '91 که در دسامبر ۱۹۹۱ در سان‌آنتونیو، تگزاس بر پا شده بود، به نمایش در آمد.

این نمایش و قابلیتهای بالقوهٔ یک مرورگر‌اَبَرمتن (hypertext browser) نظر محققان را بخود جلب کرد، و یکی از همین افراد بنام مارک آندرسن از دانشگاه ایلینویز اولین مرورگر گرافیکی را در فوریه ۱۹۹۳ به بازار عرضه کرد - این مرورگر **موزائیک** (Mosaic) نام داشت. موزائیک چنان موفقیتی بدست آورد که یک سال بعد آندرسن شرکتی بنام نت‌اسکیپ (Netscape) ، با هدف توسعهٔ نرم‌افزارهای وب، تأسیس کرد. وقتی نت‌اسکیپ در سال ۱۹۹۵ وارد بورس شد، سرمایه‌گذاران به تصور تولد یک میکروسافت دیگر ۱/۵ میلیارد دلار بابت خرید سهام آن پول پرداخت کردند - و این همه پول (که یک رکورد محسوب می‌شد) در شرایطی پرداخت شد که نت‌اسکیپ شرکت کوچکی بود با فقط یک محصول، و حتی اعلام کرده بود که تا مدتها هیچگونه سوددهی نخواهد داشت. طی سه سال بعد نت‌اسکیپ (با مرورگر **ناویگیتور** - Navigator) و میکروسافت (با مرورگر **اینترنت‌اکسپلورر** - Internet Expolrer) درگیر جنگی بودند که به «جنگ مرورگرها» معروف شد، و در آن هر شرکت سعی می‌کرد با ارائهٔ محصول بهتر دیگری را از میدان بدر کند. در سال ۱۹۹۸، شرکت America Online نت‌اسکیپ را به قیمت ۴/۲ میلیارد دلار خرید، و به عمر کوتاه آن بعنوان یک شرکت مستقل پایان داد.

در ۱۹۹۴، CERN و M.I.T موافقتنامه‌ای برای تأسیس **کنسرسیوم WWW** (که به W3C معروف است) امضا کردند. وظیفهٔ این سازمان توسعهٔ وب، استاندارد کردن پروتکلها، و تسهیل ارتباط بین سایت‌ها بود، و برنرز-لی نیز بعنوان اولین رئیس آن انتخاب شد. از آن زمان تاکنون صدها دانشگاه و شرکت بزرگ به این کنسرسیوم پیوسته‌اند. با اینکه صدها کتاب خوب دربارهٔ وب چاپ شده، اما بهترین محل برای کسب اطلاعات دربارهٔ وب خود وب است. آدرس صفحهٔ اصلی سایت کنسرسیوم وب *www.w3.org* است. در این سایت می‌توانید صدها صفحه، سند و لینک دربارهٔ کنسرسیوم وب و فعالیتهای آن ببینید.

۷-۳-۱ بررسی ساختاری

از دید یک کاربر، وب عبارتست از مجموعه‌ای فوق‌العاده بزرگ از میلیونها سند یا **صفحهٔ‌وب** (web page) . هر صفحهٔ وب می‌تواند لینکهایی به صفحات دیگر (در سراسر دنیا) داشته باشد، که برای رفتن به این صفحات کافیست روی لینک آنها کلیک کنید. این کار را می‌توان بدفعات نامحدود تکرار کرد. ایدهٔ صفحاتی که به یکدیگر اشاره می‌کنند، و اکنون آن را با عنوان اَبَرمتن (hypertext) می‌شناسیم، مدتها قبل از اختراع اینترنت و در سال ۱۹۴۵ توسط وانوار بوش (استاد مهندسی برق دانشگاه M.I.T) ابداع شده بود.

برای دیدن صفحات وب از برنامه‌ای بنام **مرورگر** (browser) استفاده می‌کنیم؛ معروفترین مرورگرهای موجود اینترنت اکسپلورر و نت‌اسکیپ ناویگیتور هستند. مرورگر بعد از آوردن صفحهٔ خواسته شده از وب، محتویات آنرا تفسیر کرده و با فرمت صحیح نمایش می‌دهد. در شکل ۷-۱۸ (الف) نمونه‌ای از یک صفحهٔ وب ساده را ملاحظه می‌کنید. مانند بسیاری از صفحات وب، این صفحه هم با یک عنوان ساده شروع شده، و بعد از مقداری اطلاعات (محتویات صفحه) به یک آدرس ایمیل ختم می‌شود. مرورگر لینکهای موجود در صفحه را، که به آنها اَبَرلینک (hyperlink) گفته می‌شود، بگونه‌ای از سایر قسمتهای متن متمایز می‌کند (با زیرخط، رنگ متفاوت، و یا هر دو). برای دنبال کردن یک لینک، کرسر ماوس را روی لینک موردنظر برده (که معمولاً با اینکار شکل کرسر عوض

می‌شود)، و کلیک کنید. با اینکه مرورگرهای کاملاً متنی هم وجود دارد (مانند لینکس – Lynx)، اما بعلت محبوبیت فراگیر مرورگرهای گرافیکی در ادامهٔ این بخش فقط دربارهٔ آنها صحبت خواهیم کرد. (اخیراً مرورگرهای کلامی نیز توسعه داده شده‌اند، که به دستورات شفاهی کاربر عکس‌العمل نشان می‌دهند.)

WELCOME TO THE UNIVERSITY OF EAST PODUNK'S WWW HOME PAGE

- Campus Information
  - ▢
  - ▢
  - ▢
  - ▢
- Academic Departments
  - ▢
  - ▢
  - ▢
  - ▢
  - ▢

Webmaster@eastpodunk.edu

(الف)

THE DEPARTMENT OF ANIMAL PSYCHOLOGY

-
- Personnel
  - ▢
  - ▢
  - ▢
-
-
- Our most popular courses
  - ▢
  - ▢
  - ▢
  - ▢
-

Webmaster@animalpsyc.eastpodunk.edu

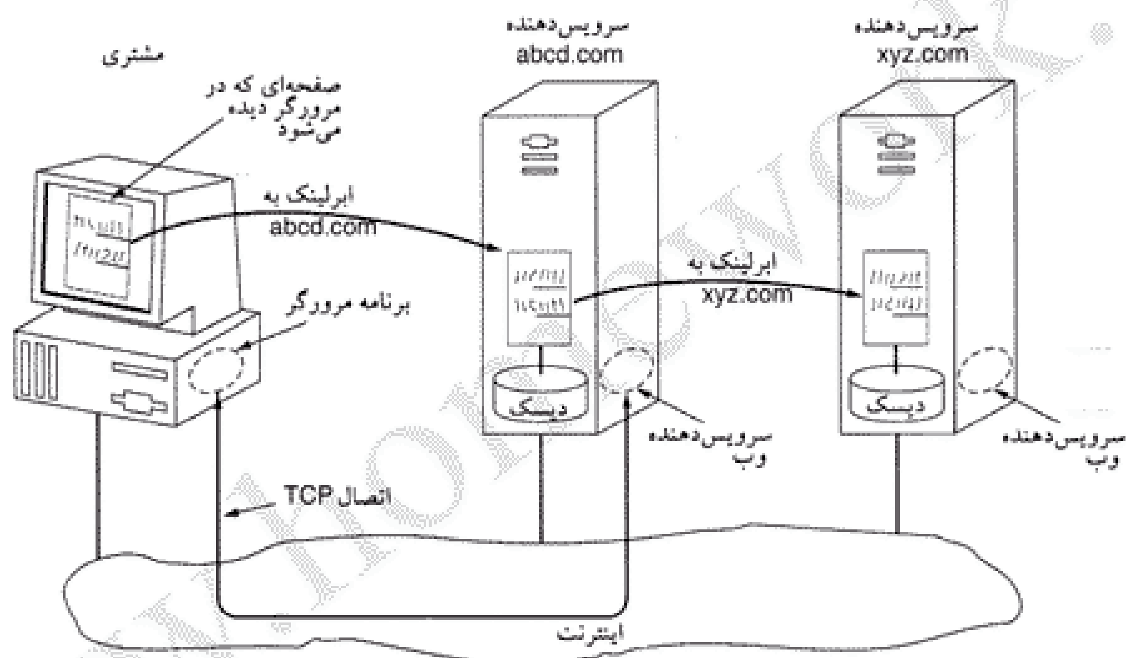
(ب)

شکل ۷-۱۸. (الف) یک صفحهٔ وب. (ب) وقتی کاربر روی لینک Department of Animal Psychology کلیک کند، به اینجا خواهد رسید.

اگر به روانشناسی حیوانات علاقمند باشید، می‌توانید روی لینک Department of Animal Psychology کلیک کنید تا به این صفحه برسید. با این کار، مرورگر صفحهٔ مشخص شده را از سایت وب آورده و نمایش می‌دهد (شکل ۷-۱۸ ب). در این صفحه هم جملاتی که زیرخط‌دار دارند لینک هستند، و با کلیک کردن آنها می‌توان به صفحات دیگر رفت. صفحه‌ای که یک لینک به آن اشاره می‌کند، می‌تواند روی همان ماشین باشد یا در

کامپیوتری آن سوی دنیا - حدس آن برای کاربر ممکن نیست، چون مرورگر بدون کمک وی صفحات را می‌آورد. اگر بعد از دیدن چند صفحه به صفحهٔ اصلی برگردید، لینکهایی که دنبال شده‌اند را با رنگ دیگری مشاهده خواهید کرد. دقت کنید که با کلیک کردن جملهٔ *Campus Information* در صفحهٔ اصلی هیچ اتفاقی نخواهد افتاد، چون این یک لینک نیست (زیرخط ندارد).

طرز کار مدل وب در شکل ۷-۱۹ نشان داده شده است. در اینجا، مرورگر در حال نمایش یک صفحه روی کامپیوتر مشتری است. وقتی کاربر روی یک جمله که به صفحه‌ای روی ماشین *abcd.com* لینک شده کلیک می‌کند، مرورگر لینک را دنبال کرده و از ماشین *abcd.com* می‌خواهد که آن صفحه را برایش بفرستد. اگر در همین صفحه لینک دیگری به یک صفحه در ماشین *xyz.com* وجود داشته باشد و کاربر آنرا کلیک کند، مرورگر درخواست خود را به سرویس‌دهنده *xyz.com* می‌فرستد (و الی آخر).



شکل ۷-۱۹. بخشی از مدل وب.

سمت مشتری

اجازه دهید فعل و انفعالات **سمت‌مشتری** (client-side) را با جزئیات بیشتر بررسی کنیم. یک مرورگر در واقع برنامه‌ایست که می‌تواند حرکات ماوس و کلیک‌های آنرا تشخیص داده و صفحات وب را نمایش دهد. وقتی یک آیتم انتخاب می‌شود، مرورگر اَبَرلینک را دنبال کرده و صفحهٔ وب را می‌آورد. بنابراین، اَبَرلینک‌ها باید راهی برای نامگذاری و مشخص کردن صفحات وب داشته باشند. صفحات وب با استفاده از URL (**یابندهٔ همسان‌منابع** - Uniform Resource Locator) نامگذاری می‌شوند. در زیر یک URL نوعی را می‌بینید:

http://www.abcd.com/products.html

در قسمتهای آینده بیشتر دربارهٔ URLها توضیح خواهیم داد. فعلاً کافیست بدانید که، یک URL سه بخش دارد: نام پروتکل (*http*)، نام DNS ماشینی که صفحه روی آن قرار دارد (*www.abcd.com*)، و نام فایل صفحهٔ وب (*products.html*).

وقتی کاربر روی اَبَرلینک کلیک می‌کند، مرورگر بایستی مراحلی را برای آوردن صفحهٔ وب طی کند. فرض

کنید کاربر بعد از کمی گشت و گذار در وب به لینک صفحهٔ اصلی ITU (که URL آن *http://www.itu.org/home/index.html* است) رسیده و می‌خواهد آنرا ببیند. اجازه دهید ببینیم بعد از انتخاب این لینک، مرورگر چه کارهایی انجام می‌دهد.

۱. مرورگر با بررسی آیتم انتخاب شده، URL آنرا بدست می‌آورد.
۲. مرورگر از DNS خود آدرس IP سایت *www.itu.org* را می‌پرسد.
۳. DNS آدرس 156.106.192.32 را برمی‌گرداند.
۴. مرورگر یک اتصال TCP به پورت 80 ماشین 156.106.192.32 برقرار می‌کند.
۵. سپس روی این لینک درخواستی برای فایل */home/index.html* به آن می‌فرستد.
۶. سرویس‌دهنده *www.itu.org* فایل */home/index.html* را به مرورگر می‌گرداند.
۷. اتصال TCP قطع می‌شود.
۸. مرورگر متن صفحهٔ */home/index.html* را نمایش می‌دهد.
۹. مرورگر تصاویر صفحه را هم آورده و نمایش می‌دهد.

اغلب مرورگرها مراحل کار خود را در خط وضعیت (پائین پنجرهٔ مرورگر) نشان می‌دهند. بدین‌ترتیب در مواقعی که مشکلی پیش می‌آید، کاربر می‌تواند ببیند که کدام مرحله مشکل‌ساز شده است (DNS دیر جواب می‌دهد، سرویس‌دهنده www سرش شلوغ است، یا ترافیک شبکه زیاد است).

برای آن که مرورگر بتواند صفحات وب را نمایش دهد، باید فرمت آنها را بداند. بهمین دلیل تمام صفحات وب با یک زبان استاندارد بنام HTML نوشته می‌شوند، تا هر مرورگری بتواند آنها را تفسیر کند. در ادامه دربارهٔ زبان HTML بیشتر صحبت خواهیم کرد.

با اینکه مرورگرها اساساً چیزی نیستند جز مفسر فایلهای HTML ، اغلب آنها دکمه‌های دیگری نیز دارند که به کاربر در **وبگردی** (web surfing) کمک می‌کند. دکمه‌های عقب (Back – برای برگشت به صفحهٔ قبلی)، جلو (Forward – برای برگشت به صفحهٔ بعدی) و خانه (Home – برای برگشت به صفحهٔ شروع) در اغلب مرورگرها وجود دارد. بسیاری از مرورگرها دارای دکمه‌ها یا منوهایی برای علامتگذاری صفحات (بمنظور ردیابی سریع آنها) نیز هستند. ذخیره و چاپ صفحات وب از دیگر امکانات مرورگرهاست، و در ضمن کاربر می‌تواند محیط کار مرورگر را مطابق نیاز خود تنظیم کند.

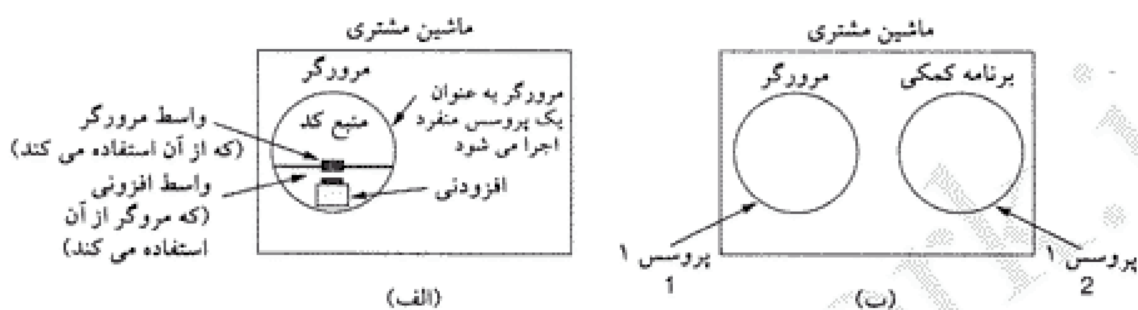
صفحات وب علاوه بر متن معمولی و آبرلینک، می‌توانند آیکون، طرح، و تصویر نیز داشته باشند، که هر یک از آنها می‌تواند به صفحات دیگر لینک شده باشد (و با کلیک کردن آنها، مرورگر صفحهٔ مشخص شده را باز خواهد کرد). حتی در برخی موارد قسمتهای مختلف یک تصویر می‌تواند به صفحات مختلفی لینک شده باشد.

تمام صفحات وب HTML نیستند؛ یک صفحهٔ وب می‌تواند حاوی سند PDF ، تصاویری با فرمتهای GIF و JPEG ، موسیقی با فرمت MP3 ، ویدئو با فرمت MPEG ، و یا صدها نوع فایل دیگر باشد. از آنجائیکه یک صفحهٔ استاندارد HTML می‌تواند لینکهایی به هر کدام از این فایلها داشته باشد، مرورگر در برخورد با صفحاتی که فرمت آنها را نمی‌شناسد، مشکل خواهد داشت.

بجای اینکه هر روز مرورگرهای بزرگتری بسازیم که بتوانند فرمتهای جدید را بخوانند، اکثر مرورگرها از روش کلی‌تر و مؤثرتری استفاده می‌کنند. وقتی سرویس‌دهنده وب صفحه‌ای را برمی‌گرداند، اطلاعات اضافه‌ای را نیز دربارهٔ آن می‌دهد، که این اطلاعات شامل نوع MIME صفحه هم می‌شود (شکل ۷–۱۲ را ببینید). صفحات *text/html* (و چند نوع سادهٔ دیگر) مستقیماً نمایش داده می‌شوند. اگر فرمت فایل یکی از این انواع شناخته شده نباشد، مرورگر برای نمایش آن با جدول انواع MIME مشورت می‌کند. در این جدول برای هر نوع MIME یک

نمایش‌دهنده (viewer) معرفی شده است.

نمایش‌دهنده‌ها بر دو نوعند: افزودنی‌ها، و برنامه‌های کمکی. **افزودنی** (plug-in) یک قطعه کُد است، که مرورگر آنرا از روی دیسک خوانده و به قابلیتهای خود اضافه می‌کند (شکل ۷-۲۰ الف). از آنجائیکه افزودنی در داخل مرورگر اجرا می‌شود، به صفحهٔ وب دسترسی داشته و می‌تواند ظاهر آنرا عوض کند. بعد از آن که افزودنی کارش را انجام داد (معمولاً بعد از اینکه کاربر به صفحهٔ دیگری رفت)، از حافظهٔ مرورگر پاک می‌شود.



شکل ۷-۲۰. (الف) یک افزودنی. (ب) یک برنامهٔ کمکی.

هر مرورگر دارای مجموعه‌ای از روالهاست که افزودنی باید آنها را پیاده‌سازی کند تا مرورگر بتواند با آن تماس برقرار کند. برای مثال، روالی باید وجود داشته باشد که مرورگر بتواند از طریق آن داده‌ها را به افزودنی بدهد. به این مجموعه از روالها **واسط افزودنی** (plug-in interface) گفته می‌شود، و برای هر مرورگر باید واسطی خاص نوشته شود. علاوه بر آن، مرورگر نیز تعدادی از روالهای خود را در اختیار افزودنی می‌گذارد تا بتواند به آن سرویس بدهد (روالهای تخصیص حافظه، نمایش وضعیت، و گرفتن پارامترها از این دسته‌اند). به این روالها **واسط مرورگر** (browser interface) می‌گویند.

قبل از استفاده از افزودنی باید آنرا نصب کرد، و این کار بر عهدهٔ کاربر است (که به سایت وب افزودنی رفته، آنرا بار کرده و نصب کند). در ویندوز، این فایل معمولاً یک فایل فشردهٔ خود-استخراج (با پسوند *exe.*) است. وقتی کاربر روی این فایل دو-کلیک کند، برنامهٔ کوچکی که در دل فایل فشرده تعبیه شده، اجرا شده و بعد از باز کردن فایل افزودنی آنرا در دایرکتوری افزودنی‌های مرورگر کپی می‌کند. سپس ارتباط ارگانیک نوع MIME افزودنی با آن را برقرار می‌کند. در سیستمهای یونیکس، این فرآیند اغلب بر عهدهٔ یک **اسکریپت پوسته** (shell script) گذاشته می‌شود.

روش دیگر گسترش دادن قابلیتهای مرورگر استفاده از **برنامهٔ کمکی** (helper application) است؛ این یک برنامهٔ کامل است که بعنوان یک پروسس مستقل اجرا می‌شود (شکل ۷-۲۰ ب). از آنجائیکه برنامهٔ کمکی یک برنامهٔ مستقل است، هیچ واسطی در اختیار مرورگر نمی‌گذارد و از سرویسهای آن هم استفاده نمی‌کند. بجای آن، نام فضای موقتی که فایل در آن ذخیره شده را از مرورگر گرفته، محتویات صفحه را خوانده و نمایش می‌دهد. برنامه‌های کمکی معمولاً برنامه‌های بزرگ و مستقلی مانند Adobe Acrobat Reader (برای نمایش فایلهای PDF) و Microsoft Word (برای نمایش فایلهای DOC) هستند. برخی از برنامه‌های کمکی برای اجرا شدن به یک افزودنی کوچک نیاز دارند.

اغلب برنامه‌های کمکی از نوع MIME *application* استفاده می‌کنند. در این نوع MIME زیرنوع‌های بسیاری تعریف شده، که *application/pdf* (برای فایلهای PDF) و *application/msword* (برای فایلهای DOC) از آن جمله‌اند. بدین ترتیب، یک URL می‌تواند مستقیماً به فایل PDF یا DOC اشاره کند، و وقتی کاربر چنین

لینکی را کلیک کند، Acrobat یا Word بطور خودکار اجرا شده و محتویات فایل را نمایش می‌دهند. با این روش می‌توان بدون کوچکترین تغییر در مرورگر، کاری کرد که بتواند انواع نامحدودی از فایلها را نمایش دهد. اغلب سرویس‌دهنده‌های وب برای خواندن صدها نوع/زیرنوع MIME پیکربندی شده‌اند، که روز به روز هم بر تعداد آنها افزوده می‌شود. البته برنامه‌های کمکی به نوع *application* محدود نیستند؛ برای مثال، برنامهٔ Adobe Photoshop از زیرنوع *image/x-photoshop* و برنامهٔ RealOne Player از زیرنوع audio/mp3 استفاده می‌کنند.

وقتی برنامه‌ای در ویندوز نصب می‌شود، خود را برای نمایش فایل MIME وابسته ثبت می‌کند. این مکانیزم وقتی چند برنامه برای نمایش یک نوع فایل (مثلاً، *video/mpg*) وجود داشته باشد، می‌تواند مشکل‌ساز شود (معمولاً برنامه‌ای که آخر نصب شده نمایش فایل را به انحصار خود در می‌آورد). در نتیجه، نصب برنامه‌های جدید می‌تواند رفتار مرورگر را در نمایش فایلها تغییر دهد.

در یونیکس، معمولاً این فرآیند خودکار نیست، و این کاربر است که فایلهای پیکربندی را به روز درمی‌آورد. این روش کار بیشتری می‌برد، ولی دردسر آن هم کمتر است.

مرورگرها (علاوه بر صفحات وب) فایلهای محلی را نیز می‌توانند باز کنند، اما از آنجائیکه این فایلها دارای نوع MIME نیستند، مرورگر باید راهی برای تشخیص برنامهٔ کمکی یا افزودنی نمایش‌دهندهٔ آنها داشته باشد. برای این کار معمولاً از پسوند نام فایل استفاده می‌شود. برخی مرورگرها (علاوه بر نوع MIME و پسوند فایل) از محتویات خود فایل نیز برای تشخیص نوع آن استفاده می‌کنند. برای مثال، اینترنت اکسپلورر بیش از نوع MIME به پسوند فایل متکی است.

در این حالت هم برنامه‌های مختلفی که قادر به نمایش یک نوع فایل هستند، بر سر باز کردن آن با هم رقابت می‌کنند. در برنامه‌های حرفه‌ای کاربر می‌تواند وابستگی برنامه به انواع MIME را فعال یا غیرفعال کند. اما برنامه‌های زیادی هم هستند که هیچ اهمیتی به این مسائل نمی‌دهند، و خیلی ساده وابستگی فایل را به خود اختصاص می‌دهند.

توانایی مرورگر در نمایش انواع مختلف فایلها بسیار جالب است، اما در عین حال می‌تواند دردسرساز باشد. برای مثال، وقتی اینترنت اکسپلورر یک فایل *exe* را می‌آورد، متوجه می‌شود که این فایل یک برنامه است و هیچ برنامهٔ کمکی برای آن ندارد: محتملترین گزینه اجرای این برنامه است. اما این می‌تواند یک رخنهٔ امنیتی بزرگ باشد. تنها کاری که یک سایت وب خرابکار باید انجام دهد، پنهان کردن لینک کُد ویروس پشت تصویری از یک ستارهٔ سینما یا قهرمان ورزشی است. کاربر از همه جا بی خبر روی تصویر موردعلاقه‌اش کلیک می‌کند، و با این کار ویروس مخرب را وارد کامپیوتر خود کرده و اجرا می‌کند. البته می‌توان اینترنت اکسپلورر را بگونه‌ای تنظیم کرد که هر برنامه‌ای را بطور خودکار اجرا نکند (و یا حداقل قبل از اجرای برنامه تأئید کاربر را بگیرد)، ولی بسیاری از کاربران مبتدی نمی‌دانند چگونه باید چنین تنظیمی را انجام دهند.

در سیستمهای یونیکس هم این اتفاق می‌تواند بیفتد، مشروط بر اینکه کاربر آگاهانه پوسته را بعنوان یک برنامهٔ کمکی تعریف کرده باشد. خوشبختانه، این کار بقدری پیچیده است که احتمال اینکه تصادفی چنین اتفاقی بیفتد، بسیار ناچیز است (در واقع، فقط عدهٔ کمی هستند که می‌توانند چنین کاری را انجام دهند).

سمت سرویس‌دهنده

کمی هم از فعل و انفعالات سمت سرویس‌دهنده بگوئیم. همانطور که قبلاً دیدید، وقتی کاربر روی یک URL (یا اَبَرلینک) کلیک می‌کند، مرورگر هر آنچه را که بین *//:http* و / بعدی قرار دارد یک نام DNS تلقی کرده، و بدنبال آدرس IP آن می‌رود. بعد از بدست آوردن آدرس IP سرویس‌دهنده، مرورگر یک اتصال TCP به پورت 80 آن

برقرار می‌کند و بقیهٔ URL را (که نام فایل صفحهٔ وب است) به سرویس‌دهنده می‌فرستد؛ سرویس‌دهنده هم در جواب صفحهٔ خواسته شده را به مرورگر برمی‌گرداند.

یک سرویس‌دهنده وب (با کمی تسامح) شبیه سرویس‌دهنده شکل ۶-۶ است. در هر دوی آنها مراحلی که سرویس‌دهنده (در حلقهٔ اصلی خود) طی می‌کند، مانند زیر است:

۱. اتصال TCP را از مشتری (مرورگر) قبول می‌کند.
۲. نام فایل درخواست شده را می‌گیرد.
۳. فایل را (از روی دیسک) می‌خواند.
۴. فایل را به مشتری برمی‌گرداند.
۵. اتصال TCP را قطع می‌کند.

سرویس‌دهنده‌های وب مدرن ویژگیهای بسیار بیشتری دارند، ولی وظیفهٔ اصلی آنها همان است که در بالا گفتیم.

یکی از مشکلات روش فوق اینست که سرویس‌دهنده برای هر درخواست باید به دیسک مراجعه کند. در نتیجه تعداد درخواستهایی که یک سرویس‌دهنده وب می‌تواند پاسخ دهد، به توانایی آن در مراجعه به دیسک بستگی دارد. زمان دسترسی در جدیدترین دیسکهای SCSI در حدود 5 msec است، و این یعنی حداکثر 200 درخواست در ثانیه (که البته اگر فایل بزرگ باشد، کمتر هم خواهد شد). این مقدار حتی برای یک سرویس‌دهنده وب متوسط هم بسیار کم است.

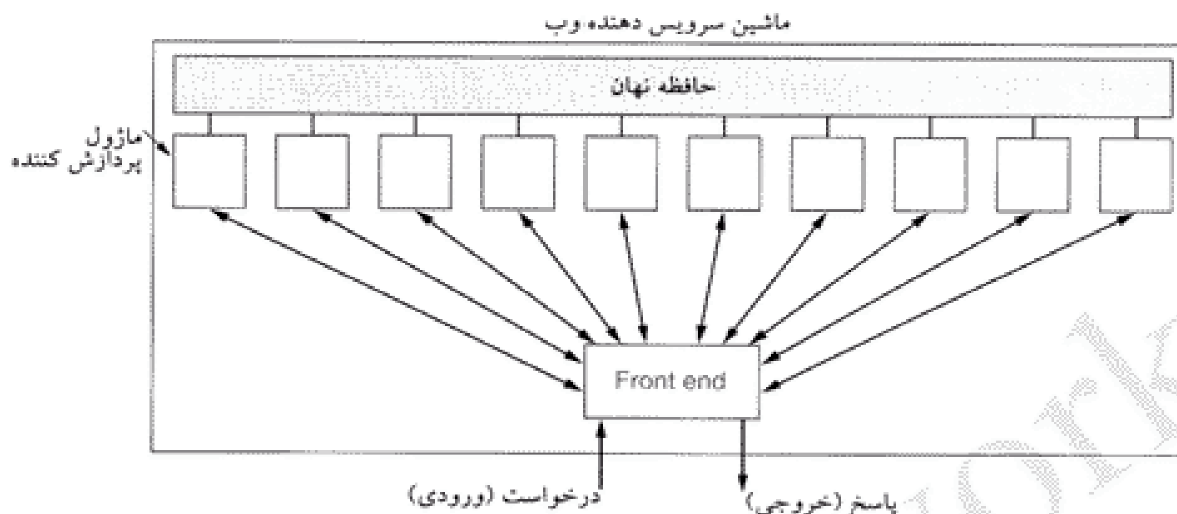
ساده‌ترین روش بهبود پاسخ سرویس‌دهنده وب (که در تمام سرویس‌دهنده‌های وب هم از آن استفاده می‌شود)، نگهداری آخرین n فایل درخواست شده در حافظهٔ RAM است. سرویس‌دهنده وب قبل از رفتن به سراغ دیسک، این **حافظهٔ نهان** (cache) را چک می‌کند. اگر فایل خواسته شده در حافظهٔ نهان باشد، از همان جا برداشته شده و به مشتری تحویل داده می‌شود (و دسترسی دیسک انجام نخواهد شد). با اینکه این تکنیک به حافظهٔ زیادی نیاز دارد و زمانی هم صرف جستجوی حافظهٔ نهان می‌شود، اما تقریباً همیشه سریعتر از دسترسی مستقیم دیسک است.

قدم بعدی برای سریعتر کردن سرویس‌دهنده وب، طراحی آن بصورت **چندریسمانی** (multithreaded) است. در این تکنیک سرویس‌دهنده وب یک ماژول **جلودار** (front-end - که درخواستهای رسیده را می‌پذیرد) و k ماژول **پردازش‌کننده** (processing) دارد (شکل ۷-۲۱ را ببینید). تمام این $k + 1$ ریسمان متعلق به یک پروسس هستند، بنابراین به فضای آدرس آن دسترسی دارند. وقتی یک درخواست جدید به سرویس‌دهنده وب می‌رسد، ماژول جلودار آن را پذیرفته، مشخصات آنرا ثبت می‌کند، و سپس به یکی از ماژولهای پردازش‌کننده تحویل می‌دهد. (تکنیک دیگری نیز وجود دارد که در آن ماژول جلودار حذف شده، و ماژولهای پردازش‌کننده مستقیماً درخواستها را دریافت می‌کنند. اما در این روش بایستی مکانیزمی برای جلوگیری از تداخل پروسسها در نظر گرفته شود.)

ماژول پردازش‌کننده ابتدا حافظهٔ نهان را چک می‌کند تا ببیند که فایل موردنظر در آنجا هست یا خیر. اگر آنجا باشد، اشاره‌گری را که به آن فایل اشاره می‌کند، به روز در می‌آورد. اگر در حافظهٔ نهان نباشد، سراغ دیسک می‌رود و مقداری از آن را در حافظهٔ نهان کپی می‌کند (که باحتمال زیاد مقداری از محتویات قدیمی حافظهٔ نهان را از بین می‌برد). بعد از کپی کردن فایل در حافظهٔ نهان، آنرا برای مشتری می‌فرستد.

مزیت این روش آن است که در زمانهایی که یکی از پروسسها در حال خواندن دیسک است (و کاری با CPU ندارد)، سایر پروسسها می‌توانند با استفاده از حافظهٔ نهان به درخواستهای رسیده پاسخ دهند. برای بهبود واقعی کارایی در سیستمهای چندریسمانی، لازم است که ماشین سرویس‌دهنده وب بجای یک دیسک چندین دیسک (و

چندین کنترلر دیسک) داشته باشد. با داشتن k دیسک در سرویس‌دهنده‌ای با k ریسمان، کارایی سیستم k برابر سیستمهای یک دیسکی خواهد بود.



شکل ۷-۲۱. یک سرویس‌دهنده وب چندریسمانی، با یک ماژول جلودار و چند ماژول پردازش‌کننده.

از نظر تئوری، یک سیستم تک‌ریسمانی با k دیسک نیز می‌تواند به همان میزان از کارایی دست یابد، ولی چنین سیستمی بسیار پیچیده‌تر خواهد بود، چون ریسمانی که در حال خواندن دیسک است، هیچ کار دیگری نمی‌تواند انجام دهد (در حالیکه این محدودیت در سیستمهای چندریسمانی وجود ندارد).

سرویس‌دهنده‌های وب جدید فقط فایل عرضه نمی‌کنند، بلکه کارهای بیشتری (که می‌توانند بسیار پیچیده باشند) انجام می‌دهند. به همین دلیل، در اغلب سرویس‌دهنده‌ها هر ماژول پردازش‌کننده (پس از دریافت درخواست از ماژول جلودار، و بسته به نوع درخواست) مراحل مختلفی را طی می‌کند.

۱. تعیین نام صفحهٔ وب خواسته شده.
۲. احراز هویت مشتری.
۳. کنترل دسترسی مشتری.
۴. کنترل دسترسی صفحهٔ وب.
۵. چک کردن حافظهٔ نهان.
۶. آوردن صفحهٔ خواسته شده از دیسک.
۷. تعیین نوع MIME فایل و نوشتن آن در پاسخ مشتری.
۸. انجام جنبه‌های دیگر درخواست.
۹. برگرداندن پاسخ به مشتری.
۱۰. نوشتن یک رکورد در دفتر ثبت وقایع (log) سرویس‌دهنده.

شاید فکر کنید مرحلهٔ ۱ اضافیست، چون نام صفحهٔ وب همیشه در درخواست مشتری قید می‌شود، اما موارد زیادی هست که نام صفحه در درخواست مشتری وجود ندارد. برای مثال، *http://www.cs.vu.nl* یک URL معتبر است که نام صفحه ندارد. در اینجا باید یک صفحهٔ پیش‌فرض وجود داشته باشد که سرویس‌دهنده (در صورت

عدم وجود نام صفحه) از آن استفاده کند. در سرویس‌دهنده‌های وب جدید حتی می‌توان زبان پیش‌فرض (مانند انگلیسی، یا ایتالیایی) هم داشت، که صفحات وب (در صورت وجود) به این زبان برگردانده می‌شوند. بعلت وجود این مکانیزمهای پیش‌فرض، آوردن نام صفحات دیگر چندان الزامی نیست.

در مرحلهٔ ۲ هویت کاربر تأئید می‌شود. این مرحله بیشتر برای صفحاتی است که در معرض دسترسی عموم قرار ندارند، و فقط افراد خاصی می‌توانند از آنها استفاده کنند. در قسمتهای آینده یکی از روشهای احراز هویت مشتری را مورد بررسی قرار خواهیم داد.

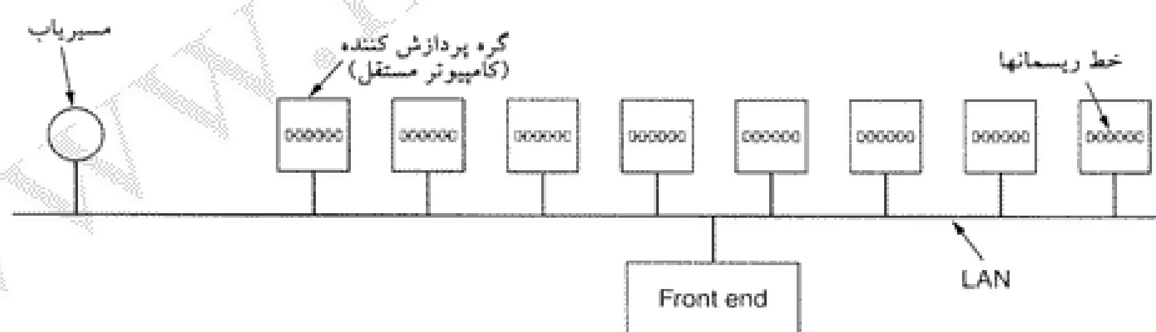
در مرحلهٔ ۳ مجوزهای کاربر در دسترسی به صفحهٔ خواسته شده چک می‌شود. مرحلهٔ بعد (مرحلهٔ ۴) این قبیل مجوزها را نسبت به خود صفحه بررسی می‌کند. مجوزهای دسترسی صفحات در فایلهای خاصی (در همان دایرکتوری که صفحه در آن قرار دارد) تعیین می‌شود - فایل *htaccess* یکی از این نوع فایلهاست.

صفحهٔ خواسته شده در مراحل ۵ و ۶ گرفته می‌شود. روتین مرحلهٔ ۶ باید بگونه‌ای طراحی شود که قادر به کار همزمان با چندین دیسک باشد.

در مرحلهٔ ۷ نوع MIME فایل (از روی پسوند فایل، عبارتی در ابتدای فایل، یا از طریق فایلهای پیکربندی) تعیین می‌شود. در مرحلهٔ ۸ کارهای متفرقه‌ای از قبیل ایجاد پروفایل کاربر یا جمع‌آوری داده‌های آماری، صورت می‌گیرد.

صفحه در مرحلهٔ ۹ برای مشتری فرستاده شده، و در مرحلهٔ ۱۰ اقدامات انجام شده در یک فایل ثبت می‌شود (که این اطلاعات معمولاً بدرد سرپرست سایت می‌خورد).

اگر تعداد درخواستهای رسیده در واحد زمان بسیار زیاد باشد، CPU (صرفنظر از تعداد دیسکهای موازی) نمی‌تواند از عهدهٔ انجام آنها برآید. راه حل این مشکل استفاده از کامپیوترهای متعدد برای توزیع بار سرویس‌دهنده است. این مدل را که **مزرعهٔ سرویس‌دهنده** (server farm) نام دارد، در شکل ۷-۲۲ ملاحظه می‌کنید. در این روش هم یک ماژول جلودار وجود دارد که درخواستها را گرفته، ولی این بار (بجای پروسسهای مختلف) در گره‌ها یا کامپیوترهای مختلف - که هر کدام می‌توانند ماشینهای چندریسمانی با چندین دیسک باشند - توزیع می‌کند.

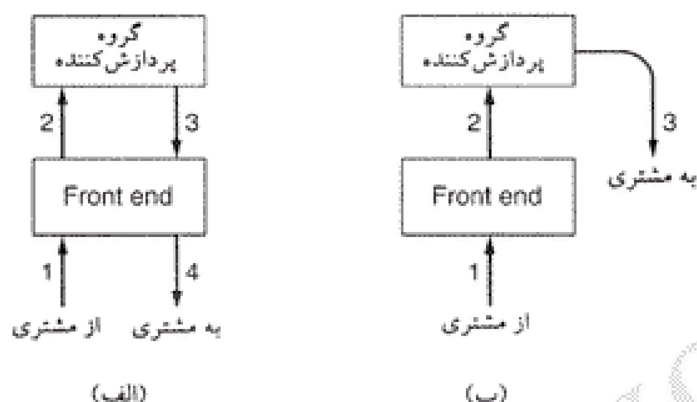


شکل ۷-۲۲. مزرعهٔ سرویس‌دهنده.

یکی از مشکلات مزرعهٔ سرویس‌دهنده این است که (بعلت جدا بودن حافظهٔ کامپیوترها) حافظهٔ نهان واحد نمی‌تواند وجود داشته باشد - مگر اینکه از کامپیوتری با چندین CPU و حافظهٔ مشترک استفاده کنیم. برای غلبه بر این مشکل و بالا بردن کارایی سیستم، ماژول جلودار می‌تواند مسیر درخواستهای مختلف را ثبت کرده و درخواستهای بعدی مرتبط با یک صفحهٔ مشخص را به همان کامپیوتر قبلی بفرستد. این تکنیک باعث می‌شود تا هر گره (کامپیوتر) به ارائهٔ صفحات خاصی اختصاص یابد، و از هرز رفتن حافظهٔ نهان جلوگیری شود.

مشکل دیگر مزرعهٔ سرویس‌دهنده این است که اتصالهای TCP به ماژول جلودار می‌رسند، و پاسخ نیز باید از همان مسیر برگردد. به شکل ۷-۲۳ (الف) نگاه کنید: در اینجا درخواست ورودی (1) و پاسخ سرویس‌دهندهٔ وب

(4) هر دو از ماژول جلودار عبور کرده‌اند. برای دور زدن این مشکل، گاهی از تکنیکی بنام پاس‌دادن **TCP** (TCP handoff) استفاده می‌شود. در این روش اتصال TCP مشتری نیز به گره پردازش‌کننده منتقل می‌شود، تا این گره بتواند پاسخ را مستقیماً به مشتری برگرداند - مسیر (3) در شکل ۷-۲۳ (ب). کل فرآیند پاس دادن TCP از دید مشتری مخفی است.



شکل ۷-۲۳. (الف) فرآیند عادی درخواست-پاسخ. (ب) درخواست-پاسخ با استفاده از تکنیک پاس دادن TCP .

URL

بارها گفته‌ایم که یک صفحهٔ وب می‌تواند لینکهایی به صفحات دیگر داشته باشد؛ اکنون خواهید دید که این کار چگونه صورت می‌گیرد. وقتی وب اختراع می‌شد، بلافاصله روشن شد که صفحات وب باید مکانیزمی برای نامگذاری و مکان‌یابی صفحاتی که به آنها لینک دارند، داشته باشند. این مکانیزم، بویژه، باید به سه سئوال اساسی دربارهٔ صفحهٔ انتخاب شده جواب دهد:

۱. نام این صفحه چیست؟
۲. کجا قرار دارد؟
۳. چگونه می‌توان آنرا خواند؟

اگر هر صفحه دارای یک نام منحصربفرد باشد، پیدا کردن آن قاعدتاً نباید مشکلی داشته باشد. اما این راه حل هم مشکل ما را برطرف نخواهد کرد. با یک مثال موضوع روشنتر خواهد شد. در ایالات متحدهٔ آمریکا هر فرد یک شمارهٔ تامین اجتماعی منحصربفرد دارد، اما آیا فقط با داشتن این شماره می‌توان فرد موردنظر را پیدا کرد. آدرس این فرد کجاست؟ با چه زبانی باید با او تماس گرفت؟ همین مسائل برای وب هم مصداق دارند.

راه حلی که برای این موضوع پیدا شد، به هر سه سئوال فوق یکجا جواب می‌دهد. در این راه حل، هر صفحه وب یک URL (یابندهٔ‌همسان‌منابع - Uniform Resource Locator) دارد، که آنرا بگونه‌ای منحصربفرد در تمام دنیا مشخص می‌کند. هر URL سه بخش دارد: پروتکل (سئوال ۳)، نام DNS ماشینی که صفحه روی آن قرار دارد (سئوال ۲)، و نام صفحه در آن ماشین (سئوال ۱). بعنوان نمونه در سایت وب مؤلف کتاب چندین فایل ویدئویی از شهر آمستردام و دانشگاه آن وجود دارد، که در صفحه‌ای با URL زیر قرار دارند:

http://www.cs.vu.nl/video/index-en.html

سه بخش این URL عبارتند از: پروتکل (*http*) ، نام DNS ماشین سرویس‌دهنده (*www.cs.vu.nl*) ، و نام صفحهٔ وب (*video/index-en.html*) که با / از هم جدا شده‌اند. نام صفحهٔ وب عبارتست از مسیر نسبی فایل در دایرکتوری وب پیش‌فرض کامپیوتر *cs.vu.nl* .

در اغلب سایتهای وب از نامهای کوتاه شده برای فایلها استفاده می‌شود. اگر نام فایل در URL وجود نداشته باشد، معمولاً صفحهٔ اصلی سایت (home page) برگردانده می‌شود. اگر فقط دایرکتوری صفحه مشخص شده باشد، سرویس‌دهنده وب فایل پیش‌فرض آن دایرکتوری (که معمولاً *index.html* نام دارد) را به مشتری برمی‌گرداند. در برخی از سرویس‌دهنده‌های وب نیز از نامهای کوتاهی مانند *~user/* برای مشخص کردن دایرکتوری اختصاصی اشخاص استفاده می‌شود. بعنوان مثال، برای دسترسی به صفحهٔ اصلی فضای وب مؤلف می‌توانید از URL زیر استفاده کنید:

http://www.cs.vu.nl/~ast/

اکنون اجازه دهید ببینیم اَبرمتن (hypertext) چگونه کار می‌کند. برای اینکه یک عبارت قابل کلیک باشد، نویسندهٔ متن باید دو چیز را مشخص کند: متنی که باید دیده شود، و URL صفحه‌ای که بعد از کلیک شدن عبارت باید باز شود (بعداً در همین فصل روش ایجاد چنین عبارتهایی را خواهید دید). وقتی این عبارت کلیک شود، مرورگر آدرس سرویس‌دهنده صفحه را با استفاده از DNS پیدا کرده، یک اتصال TCP به آن برقرار می‌کند و نام فایل و پروتکل را به سرویس دهنده می‌دهد. سرویس‌دهنده هم صفحهٔ خواسته شده را برمی‌گرداند.

یکی از ویژگیهای URL این است که می‌توان براحتی پروتکل مورد استفاده را عوض کرده و به منابع مختلف دسترسی پیدا کرد. در حقیقت تعداد زیادی از این پروتکلها تعریف شده، که برخی از معروفترین آنها را در شکل ۷-۲۴ ملاحظه می‌کنید.

نامه	کاربرد	مثال
http	ابرمتن (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	فایل محلی	file:///usr/suzanne/prog.c
news	گروه خبری	news:comp.os.minix
news	مقاله خبری	news:AA0134223112@cs.utah.edu
gopher	گوفر	gopher://gopher.tc.umn.edu/11/Libraries
mailto	ارسال ایمیل	mailto:JohnUser@acm.org
telnet	ورود از راه دور	telnet://www.w3.org:80

شکل ۷-۲۴. چند URL معروف.

اجازه دهید این فهرست را مختصراً بررسی کنیم. پروتکل *http* زبان وب است، زبانی که سرویس‌دهنده‌های وب با آن صحبت می‌کنند. HTTP مخفف HyperText Transfer Protocol (**پروتکل انتقال اَبرمتن**) است. بعداً دربارهٔ این پروتکل بیشتر صحبت خواهیم کرد.

پروتکل *ftp* (**پروتکل انتقال فایل** – File Transfer Protocol) برای انتقال فایل در اینترنت مورد استفاده قرار می‌گیرد. FTP بیش از دو دهه است که در اینترنت حضور دارد، و پروتکلی کاملاً جا افتاده است. در سراسر دنیا تعداد زیادی سرویس‌دهنده FTP وجود دارد که کاربران اینترنت می‌توانند وارد آنها شده و فایلهای موجود در آنها را بار کنند. وب تغییر چندانی در این وضعیت نداد، فقط کارها را ساده‌تر کرد (چون واسط کاربر FTP کمی کهنه و قدیمی شده بود). FTP بسیار قویتر از HTTP است، چون اجازه می‌دهد کاربر ماشین *A* فایلی را از ماشین *B* به ماشین *C* منتقل کند.

مرورگرها می‌توانند با استفاده از پروتکل *file* مستقیماً با فایلها نیز (مانند صفحات وب) کار کنند، البته فقط با فایلهای محلی (روی همان کامپیوتر) نه فایلهای کامپیوترهای دیگر. روش کار شبیه FTP است، با این تفاوت که

نیازی به سرویس‌دهنده FTP ندارد.

مدتها قبل از اینکه اینترنت بدنیا بیاید، یک سیستم خبری وجود داشت بنام USENET . این سیستم میلیونها کاربر را در بیش از ۳۰,۰۰۰ **گروه‌خبری** (newsgroup) گرد هم آورده بود، کاربرانی که دربارهٔ موضوعات مختلف و متنوع بحث کرده و مقاله رد و بدل می‌کردند. برای ارتباط با این گروههای خبری می‌توان از پروتکل *news* استفاده کرد (یعنی، مرورگرها می‌توانند خبرخوان هم باشند، و در واقع بعضی از آنها حتی از برنامه‌های تخصصی خبرخوان نیز بهتر هستند).

پروتکل *news* از دو فرمت پشتیبانی می‌کند. در فرمت اول بعد از مشخص کردن گروه خبری، می‌توان از میان فهرست مقالات آن مقالهٔ مورد نظر را انتخاب کرد. در فرمت دوم شمارهٔ شناسایی مقاله را باید مشخص کرد (AA0134223112@cs.utah.edu در شکل ۷-۲۴). مرورگرها برای آوردن مقالات گروه خبری از پروتکل NNTP(**پروتکل انتقال اخبار شبکه** - Network News Transfer Protocol) استفاده می‌کنند. در این کتاب دربارهٔ NNTP صحبت نخواهیم کرد، فقط کافیست یادآور شویم که این پروتکل تا حدی شبیه SMTP است و مانند آن عمل می‌کند.

از پروتکل *gopher* در سیستمهای گوفر استفاده می‌شود؛ این سیستم در دانشگاه مینه‌سوتا اختراع شد، و نام آن از تیم ورزشی این دانشگاه (**گوفرهای طلایی** - گوفر نوعی موش خرمای بزرگ است، و در ضمن لقبی است که به اهالی ایالت مینه‌سوتا نیز داده شده) گرفته شده است. گوفر سالها قبل از وب وجود داشت، و نوعی سیستم بازیابی اطلاعات محسوب می‌شود (که متأسفانه فقط متنی است، و قابلیتهای گرافیکی ندارد). این سیستم سالهاست که منسوخ شده، و دیگر بندرت کسی از آن استفاده می‌کند.

دو پروتکل آخر شکل ۷-۲۴ ارتباطی با صفحات وب ندارند، و کاربردهای دیگری دارند. پروتکل *mailto* اجازه می‌دهد که کاربر از داخل مرورگر ایمیل بفرستد. برای این کار کافیست روی عبارتی که URL آن *mailto:somebody@abcd.com* است، کلیک کنید. اغلب مرورگرها برای ارسال ایمیل به این آدرس از برنامهٔ ایمیل پیش‌فرض سیستم استفاده می‌کنند. پروتکل *telnet* نیز برای ایجاد ارتباط با کامپیوترهای دیگر بکار می‌رود، و طرز کار آن بسیار شبیه برنامهٔ telnet است.

استفاده از پروتکلهای مختلف به کاربر اجازه می‌دهد تا یک برنامهٔ واحد (مرورگر وب) را برای کارهای مختلف بکار گیرد. اگر نمی‌دانستیم که وب و مرورگر را یک فیزیکدان اختراع کرده، قطعاً آنرا دستپخت بخش تبلیغات یک شرکت بزرگ نرم‌افزاری تصور می‌کردیم.

با وجود تمام این ویژگیهای جالب URL ، رشد سرسام‌آور وب ضعف ذاتی آن را آشکار کرد: هر URL به یک سرویس‌دهنده مشخص اشاره می‌کند. برای کاهش ترافیک صفحاتی که تعداد مراجعان آن زیاد است، بهتر است یک URL چندین سرویس‌دهنده داشته باشد. اما مشکل اینجاست که، URLها هیچ راهی ندارند که بدون ذکر آدرس دقیق صفحهٔ وب، به آن اشاره کنند. برای مثال، یک URL نمی‌تواند بگوید: «من صفحهٔ xyz را می‌خواهم، اهمیتی هم نمی‌دهم آنرا از کجا می‌آوری.» برای حل این مشکل و امکان تکثیر صفحات وب، IETF در حال کار روی سیستمی از URN (**نام جهانی منابع** - Universal Resource Name) هاست. در کل، URN را می‌توان یک URL عمومی دانست. تحقیقات در این زمینه همچنان ادامه دارد، اما پیشنهادهایی تحت RFC 2141 نیز ارائه شده است.

بدون حالتی و کوکی

همانطور که بارها خاطر نشان کردیم، وب اساساً **بدون‌حالت** (stateless) است، و چیزی مانند **ورود به سیستم** (login) در آن وجود ندارد. مرورگر درخواست خود را به سرویس‌دهنده می‌فرستد، سرویس‌دهنده هم یک فایل

به آن برمی‌گرداند، و بعد بکلی فراموش می‌کند که اصلاً چنین مشتریی را دیده است.

در اوایل کار، زمانیکه وب فقط یک محیط عمومی بود، این مدل کاملاً کفایت می‌کرد. ولی با گسترش کاربردهای وب این موضوع مشکل‌ساز شد. بعنوان مثال، برای استفاده از برخی سایتهای وب کاربران باید ثبت نام کنند، و احیاناً پولی بپردازند. این وضعیت سئوالی را پیش می‌آورد: «سرویس‌دهنده وب چگونه باید بین درخواستهای کاربرانی که ثبت‌نام کرده‌اند، و کاربران عادی فرق قائل شود؟» مثال دیگر مربوط به **تجارت‌الکترونیک** (e-commerce) می‌شود: «کاربری در یک فروشگاه الکترونیکی می‌چرخد و اجناس موردنیازش را در یک **سبد خریدالکترونیکی** (shopping cart) قرار می‌دهد؛ سئوال اینست که سرویس‌دهنده وب چگونه محتویات هر سبد خرید را ردگیری می‌کند؟» مثال سوم به سایتهایی که اجازه می‌دهند کاربر فضاهای اختصاصی داشته باشد (مانند Yahoo)، مربوط می‌شود: «سرویس‌دهنده وب چگونه کاربران را شناسایی می‌کند، و بخاطر می‌سپارد که هر کاربر صفحه‌اش را چطور تنظیم کرده است؟»

در نگاه اول شاید فکر کنید که سرویس‌دهنده وب می‌تواند از آدرس IP کاربر برای شناسایی وی استفاده کند. اما این روش کار نمی‌کند. اول اینکه، بسیاری کاربران از کامپیوترهای مشترک استفاده می‌کنند (بویژه کارمندان شرکتها)، و آدرس IP فقط بدرد شناسایی کامپیوترها می‌خورد، نه کاربران. دیگر اینکه، اغلب ISPها از تکنیک NAT استفاده می‌کنند، و تمام بسته‌هایی که از این ISPها خارج می‌شود یک آدرس IP دارند. از دیدگاه سرویس‌دهنده وب تمام کاربران این ISP یک آدرس IP بیشتر ندارند.

برای حل این مشکل، نت‌اسکیپ تکنیکی اختراع کرد بنام **کوکی** (cookie)، که انتقادات زیادی به آن وارد شد. این نام از نوعی برنامهٔ خاص گرفته شده، که کاری را انجام می‌دهد و آنرا در جایی ثبت می‌کند، تا بعدها بتواند دوباره بیاد بیاورد چه کار کرده است (در سیستم عامل هم چنین مفهومی بکرات مورد استفاده قرار می‌گیرد). کوکی بعدها در RFC 2109 جنبهٔ رسمی بخود گرفت.

وقتی مشتری یک صفحهٔ وب درخواست می‌کند، سرویس‌دهنده مقداری اطلاعات اضافی همراه آن می‌فرستد که کوکی هم می‌تواند جزئی از آن باشد. کوکی معمولاً یک فایل کوچک (حداکثر 4 KB) یا یک رشتهٔ متنی است. مرورگر این کوکی‌ها را در یک دایرکتوری خاص روی کامپیوتر مشتری ذخیره می‌کند (البته اگر کاربر این ویژگی را غیرفعال نکرده باشد). کوکی‌ها فقط فایل یا رشته‌های متنی هستند، نه برنامه‌های اجرایی. در حقیقت، کوکی حتی می‌تواند حاوی کُد یک ویروس باشد، اما از آنجاییکه کوکی‌ها اساساً داده تلقی می‌شوند، چنین ویروسی هیچ راهی برای اجرا شدن در کامپیوتر مشتری (و صدمه زدن به آن) ندارد. با این حال، همیشه احتمال آن هست که بالاخره یک هکر بتواند راهی برای این کار پیدا کند.

هر کوکی می‌تواند تا پنج فیلد داشته باشد (شکل ۷-۲۵ را ببینید). فیلد *Domain* مشخص می‌کند که کوکی از کجا آمده است. مرورگرها مکانیزمهایی دارند تا مطمئن شوند که سرویس‌دهنده دروغ نگفته است. هر ناحیه می‌تواند حداکثر ۲۰ کوکی (برای هر مشتری) در یک کامپیوتر ذخیره کند. فیلد *Path* مشخص می‌کند که کدام بخش از سیستم فایل سرویس‌دهنده وب می‌تواند از کوکی استفاده کند. این فیلد اغلب / است، که به معنای کل سیستم فایل می‌باشد.

ناحیه	مسیر	محتویات	زمان انقضا	ایمنی
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	بلی
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	خیر
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	خیر
sneaky.com	/	UserID=3627239101	31-12-12 23:59	خیر

شکل ۷-۲۵. چند نمونه کوکی.

فیلد *Content* به شکل *name = value* است، که *name* و *value* می‌توانند هر چیزی (که سرویس‌دهنده می‌خواهد) باشند. این فیلدیست که محتویات کوکی در آن ذخیره می‌شود.

فیلد *Expires* مشخص می‌کند که کوکی تا چه زمانی اعتبار دارد. اگر در این فیلد مقدار وجود نداشته باشد، مرورگر هنگام خروج کوکی را دور می‌اندازد. به این قبیل کوکی‌ها **کوکی بی‌دوام** (nonpersistent cookie) می‌گویند. اگر فیلد *Expires* کوکی تاریخ و ساعت داشته باشد، به آن **بادوام** (persistent) می‌گویند، و کوکی تا این زمان در سیستم حفظ خواهد شد. زمان انقضا بوقت گرینویچ است. اگر سرویس‌دهنده بخواهد یک کوکی را از روی سیستم مشتری پاک کند، کافیست همان کوکی را دوباره با تاریخ انقضایی در گذشته بفرستد.

و بالاخره فیلد *Secure* می‌گوید که مرورگر باید کوکی را فقط به یک سرویس‌دهنده امن (secure) پس بفرستد. این ویژگی بیشتر برای تجارت الکترونیک، عملیات بانکی و سایر کاربردهایی که به ایمنی بالا نیاز دارند، مورد استفاده قرار می‌گیرد.

دیدید که سرویس‌دهنده چطور کوکی‌ها را به مشتری می‌فرستد، اما طرز کار آنها چگونه است؟ درست قبل از اینکه مرورگر درخواستی را به یک سرویس‌دهنده وب بفرستد، دایرکتوری کوکی‌ها را چک می‌کند تا ببیند آیا ناحیهٔ مقصد در آنجا کوکی دارد یا خیر. اگر چنین باشد، مرورگر تمام آن کوکی‌ها را همراه با درخواست خود به سرویس‌دهنده وب می‌فرستد. وقتی سرویس‌دهنده این کوکی‌ها را دریافت کرد، خود می‌داند با آنها چه کند.

اجازه دهید چند نمونه از کاربرد کوکی‌ها را بررسی کنیم. در شکل ۷-۲۵، اولین کوکی متعلق به ناحیهٔ *toms-casino.com* است، و به کار شناسایی کاربر می‌آید. وقتی کاربر به خیال بردن مقداری پول وارد این سایت می‌شود، سرویس‌دهنده با استفاده از این کوکی وی را شناسایی می‌کند. برای مثال، سرویس‌دهنده وب می‌تواند به کمک این عدد مشخصات کاربر را از پایگاه دادهٔ خود استخراج کرده، و صفحه را به شکل مناسب (بسته به علائق قبلی کاربر) به نمایش در آورد.

کوکی دوم متعلق به سایت *joes-store.com* است. در این سناریو کاربر ما مشغول چرخیدن در یک فروشگاه مجازی و انتخاب کالاست. وقتی کاربر کالای مناسبی را یافت، روی آن کلیک می‌کند؛ با این کار، سرویس‌دهنده وب یک کوکی به کامپیوتر مشتری می‌فرستد، که در آن تعداد اجناس انتخاب شده و شمارهٔ آنها قید شده است. هر بار که کاربر صفحهٔ جدیدی را در این فروشگاه باز می‌کند (و در واقع درخواست جدیدی می‌فرستد)، این کوکی را هم به همراه آن به سرویس‌دهنده وب برمی‌گرداند. در مثال شکل ۷-۲۵ سه قلم جنس در سبد خرید مشتری وجود دارد، که از سومی دو عدد انتخاب شده است. در پایان هم وقتی مشتری پای صندوق می‌رود، سرویس‌دهنده با استفاده از همین کوکی می‌تواند قیمت کل خریدهای مشتری را به وی اعلام کند.

کوکی سوم متعلق به یکی از **درگاه‌های وب** (web portal) است. وقتی کاربر می‌خواهد وارد این سایت شود، مرورگر این کوکی را به همراه درخواست وی به سرویس‌دهنده وب می‌فرستد. سرویس‌دهنده وب هم صفحه‌ای به کاربر برمی‌گرداند که در آن نرخ سهام شرکتهای Sun Microsystems و Oracle ، و نتایج تیم فوتبال New Yprk Jets نشان داده شده است. از آنجائیکه یک کوکی می‌تواند تا 4 KB باشد، کاربر می‌تواند فهرست بلندبالایی از علائق خود را در این سایت ثبت کند.

یک سرویس‌دهنده وب می‌تواند از کوکی فقط برای مصارف داخلی خودش استفاده کند، مثلاً تعداد کاربرانش را بداند، و یا بداند که هر بازدیدکننده قبل از ترک سایت چند صفحه را دیده است. وقتی اولین بازدیدکننده وارد سایت می‌شود، هنوز هیچ کوکیی از آن در کامپیوترش ندارد، پس سرویس‌دهنده یک کوکی که در آن عبارت *Counter = 1* نوشته شده به مشتری می‌فرستد. کلیک بعدی کاربر باعث می‌شود تا همین کوکی به سرویس‌دهنده برگردد. سرویس‌دهنده هم مقدار *Counter* را یکی افزایش داده، و دوباره به مشتری برمی‌گرداند. بدین ترتیب

سرویس‌دهنده می‌تواند آماری از تعداد بازدیدکنندگان سایت (و اینکه هر کدام چند صفحه را دیده‌اند) بدست آورد.

امکان سوءاستفاده از کوکی‌ها نیز وجود دارد. در تئوری، مرورگر کوکی‌ها را فقط به سایتی که به آن تعلق دارند می‌فرستد، ولی برخی از هکرها توانسته‌اند با استفاده از باگهایی که در مرورگرها وجود دارد، کوکی‌هایی را که متعلق به آنها نیست بدست آورند. از آنجائیکه برخی از سایتهای تجارت الکترونیک شمارهٔ کارت اعتباری مشتریان خود را در کوکی‌ها ذخیره می‌کنند، می‌توانید خطرات بالقوهٔ این وضعیت را بروشنی دریابید.

نقطه ضعف دیگر کوکی‌ها (که بحثهای زیادی را هم بدنبال داشته) امکان استفاده از آنها برای جمع‌آوری مخفیانهٔ اطلاعات دربارهٔ کاربران و عادتهای وبگردی آنهاست. برای مثال، شرکت تبلیغاتی Sneaky Ads با سایتهای معروف تماس گرفته و از آنها می‌خواهد که (در ازای دریافت اجرت) اعلان تبلیغاتی این شرکت را بالای سایت خود نصب کنند (و همانطور که می‌دانید این بزرگترین منبع درآمد سایتهای وب است). اما Sneaky Ads بجای دادن تصویر GIF یا JPEG آگهی تبلیغاتی خود، یک URL به این شرکتها می‌دهد تا در صفحات خود قرار دهند. هر URL که این شرکت به طرفهای خود می‌دهد، یک شمارهٔ منحصربفرد دارد، مانند زیر

http://www.sneaky.com/382674902342.gif

وقتی کاربر وارد صفحهٔ یکی از این سایتها (مثلاً، صفحهٔ *P*) می‌شود، مرورگر فایل HTML آنرا می‌خواند، و بعد از آنکه متوجه شد در آن یک لینک به تصویری در سایت www.sneaky.com وجود دارد، درخواستی برای خواندن این تصویر به سایت مزبور می‌فرستد. سرویس‌دهنده این سایت به همراه تصویر آگهی، یک کوکی با شماره شناسایی منحصربفرد 3627239101 به مرورگر پس می‌فرستد (شکل ۷-۲۵ را ببینید). سپس، سایت Sneaky بازدید کاربر از صفحهٔ *P* را ثبت می‌کند (و این کاری ساده است، چون سرویس‌دهنده وب می‌داند که فایل *382674902342.gif* مربوط به صفحهٔ *P* است). البته احتمالاً آگهی همان است، فقط نام فایل آن در هر سایت فرق می‌کند.

بعدها وقتی کاربر وارد سایت دیگری که آگهی شرکت Sneaky Ads در آن قرار دارد، می‌شود مرورگر فایل HTML آنرا خوانده و سپس از سایت www.sneaky.com درخواست ارسال تصویر (مثلاً *493654919923.gif*) را می‌کند. اما از آنجائیکه مرورگر یک کوکی از سایت www.sneaky.com دارد، این کوکی را نیز همراه درخواست خود می‌فرستد؛ و اکنون Sneaky Ads دومین صفحه‌ای را که کاربر ما از آن بازدید کرده، می‌شناسد.

به مرور زمان، شرکت Sneaky Ads اطلاعات کاملی دربارهٔ تعداد زیادی از کاربران و عادتهای وبگردی آنها جمع آوری می‌کند (بدون اینکه آنها حتی یکبار روی آگهی‌های این شرکت کلیک کرده باشند!). البته این شرکت هنوز نام کاربر را نمی‌داند (اگرچه آدرس IP او را می‌داند، و همین برای بدست آوردن اطلاعات بعدی کافیست) - و اگر کاربر بیچاره حتی برای یک بار نامش را در اختیار یکی از شرکتهای همکار Sneaky Ads گذاشته باشد، که دیگر کار تمام است. امروزه فروش اطلاعات کاربران اینترنت و علائق آنها (به طالبان این قبیل اطلاعات) یکی از منابع سرشار درآمد در وب محسوب می‌شود. بدترین جنبهٔ این نوع جمع‌آوری اطلاعات آنست که کاربر حتی روحش از آن خبر ندارد، و فکر می‌کند چون روی هیچ آگهی اینترنتی کلیک نکرده کاملاً در امان است!

و اگر Sneaky Ads بخواهد پلیدی را به نهایت برساند، حتی لازم نیست یک آگهی کلاسیک بدهد؛ یک آگهی که فقط یک پیکسل (آن هم به رنگ زمینهٔ سایت!) باشد، برای کار او کافیست (مرورگر باز هم مجبور است برای این تصویر یک پیکسلی به سایت www.sneaky.com برود، و کوکی را بگیرد).

برخی از کاربران (برای حفظ حریم شخصی خود) مرورگر را طوری پیکربندی می‌کنند که تمام کوکی‌ها را رد کند. اما این کار در عملکرد سایتهای معتبری که به کوکی وابسته‌اند، نیز اختلال ایجاد می‌کند. برای حل این مشکل می‌توان از نرم‌افزارهایی که به **کوکی‌خور** (cookie-eating) معروفند، استفاده کرد. این نرم‌افزارها تمام کوکی‌هایی

را که به یک کامپیوتر می‌شوند، گرفته و فقط آنهایی را ذخیره می‌کنند که از نظر کاربر مجاز شناخته شده باشند. مرورگرهای جدید هم به کاربران امکان کنترل کامل کوکی‌ها را می‌دهند.

۷-۳-۲ سندهای وب استاتیک

اساس وب عبارتست از انتقال صفحات وب از سرویس‌دهنده به مشتری. صفحات وب، در ساده‌ترین شکل خود، استاتیک (ثابت و ایستا) هستند و فقط روی سرویس‌دهنده منتظرند تا کسی بیاید و آنها را بردارد. با این تعریف، حتی یک فیلم ویدئویی نیز استاتیک است، چون چیزی نیست جز یک فایل ساده. در این قسمت صفحات وب استاتیک را بتفصیل مورد بررسی قرار خواهیم داد.

HTML

صفحات وب با زبانی بنام HTML (زبان علامتگذاری اَبَرمتن - HyperText Markup Language) نوشته می‌شوند. با HTML می‌توان متن، گرافیک و لینک به صفحات وب اضافه کرد. HTML یک زبان علامتگذاریست، یعنی زبانی که نشان می‌دهد سند چگونه باید فرمت شود. «علامتگذاری» اصطلاحیست قدیمی که به دوران چاپ با حروف سربی برمی‌گردد، و در آن ویراستار با علامتگذاری متن نوشته به حروفچین نشان می‌داد که چگونه (با چه حروف و اندازه‌ای) باید صفحهٔ چاپی را بچیند. در زبانهای علامتگذاری فرمانهای مشخصی برای فرمت کردن سند وجود دارد. برای مثال، در HTML علامت **فرمان شروع فرمت با فونت ضخیم**، و فرمان پایان فونت ضخیم است. مزیت زبان علامتگذاری اینست که نوشتن مرورگر برای آن ساده است، چون فقط کافیست مرورگر فرمانهای علامتگذاری را بداند. زبانهای TeX و troff هم جزء زبانهای علامتگذاری معروف هستند.

با نوشتن فرمانهای علامتگذاری در فایل HTML و استاندارد کردن این فرمانها، تمام مرورگرها می‌توانند صفحات وب را خوانده و فرمت کنند. این ویژگی اهمیت بسیار زیادی در نمایش صحیح صفحات وب دارد، چون ممکنست یک صفحهٔ وب در کامپیوتری با وضوح 1200 × 1600 پیکسل و رنگ 24-bit ایجاد شده باشد، ولی کامپیوتر بازدیدکننده فقط وضوحی معادل 480 × 640 پیکسل و رنگ 8-bit داشته باشد.

در این قسمت HTML را بطور مختصر بررسی خواهیم کرد. یک سند HTML را می‌توان با هر ادیتوری نوشت، اما برای نوشتن صفحات HTML برنامه‌های خاصی نیز طراحی شده است، که امکانات بیشتری در اختیار فرد قرار می‌دهند (و در ضمن دست او را در ریزه‌کاریها می‌بندند).

هر صفحهٔ وب یک سر (head) و یک بدنه (body) دارد، که بین برچسبهای <html> و </html> قرار می‌گیرند، اگرچه اغلب مرورگرها نبودن این دو برچسب را نادیده می‌گیرند (برچسب - tag - فرمان فرمت HTML است که داخل < > نوشته می‌شود). همانطور که در شکل ۷-۲۶ (الف) می‌بینید، قسمت سر بین برچسبهای <head> و </head> ، و قسمت بدنه بین برچسبهای <body> و </body> محصور می‌شوند. دستورات HTML داخل این برچسبها نوشته می‌شود. اغلب فرمانهای HTML دارای چنین شکلی هستند، یعنی با <something> شروع، و به </something> ختم می‌شوند. در اغلب مرورگرها فرمانی وجود دارد بنام VIEW SOURCE (یا چیزی شبیه آن)، که به کمک آن می‌توان (بجای خروجی فرمت‌شده) محتویات فایل HTML را دید.

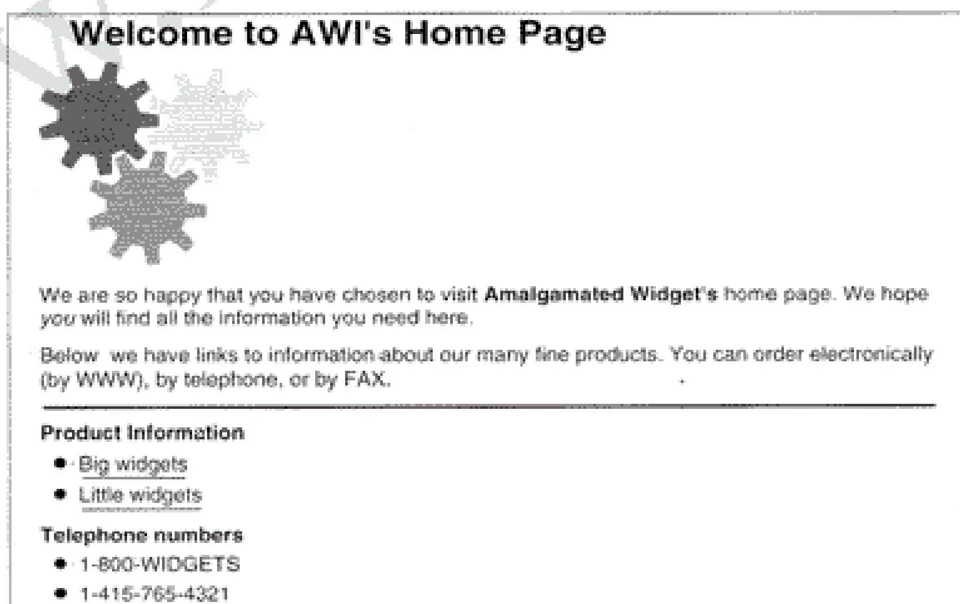
نوع حروف در برچسبهای HTML تفاوتی ندارد، بعبارت دیگر برچسبهای <head> و <HEAD> یکی هستند؛ البته در استانداردهای جدید فقط می‌توان از حروف کوچک برای نوشتن برچسبها استفاده کرد. فرمت خود فایل HTML هیچ تأثیری روی خروجی آن ندارد، چون مرورگرها مجبورند صفحات را در کامپیوترهای مختلف

نمایش دهند، و بهمین دلیل فاصله‌ها و فضاهای خالی را حذف کرده و صفحه را متناسب با تنظیمات کامپیوتر مقصد از نو فرمت می‌کنند. البته نویسندهٔ فایل HTML می‌تواند از فاصله و فضای خالی برای خواناتر کردن آن استفاده کند (که اتفاقاً چیز بسیار خوبی هم هست). در نتیجه، برای فاصله انداختن بین پاراگرافها نمی‌توانید از Enter استفاده کنید: برای این کار برچسب مخصوصی وجود دارد.

بعضی از برچسبها دارای پارامترهای نام‌دار (named parameter) هستند، که به صفت (attribute) معروفند.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a>
   <li> <a href="http://widget.com/products/little"> Little widgets </a>
</ul>
<h2> Telephone numbers </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax: 1-415-765-4321
</ul>
</body>
</html>
```

(الف)



(ب)

شکل ۷-۲۶. (الف) کد HTML برای صفحه وب نمونه (ب) صفحه قالب‌دهی شده بر روی مرورگر.

مثلاً، در فرمان

<img src="abc" alt="foobar">

برچسب <img> دو پارامتر بنامهای *src* و *alt* دارد، که مقدار آنها بترتیب *abc* و *foobar* است. در استاندارد HTML فهرستی از پارامترهای مجاز هر برچسب و مفهوم آنها داده شده است. از آنجائیکه این پارامترها نام‌دار هستند، ترتیب نوشتن آنها اهمیتی ندارد.

از نظر فنی، سندهای HTML با مجموعه کاراکتر ISO 8859-1 Latin-1 نوشته می‌شوند، ولی از آنجائیکه اغلب کاربران از صفحه کلیدهای ASCII استفاده می‌کنند، برای نمایش کاراکترهای خاص (مانند è) می‌توان از **توالی گریز** (escape sequence) کمک گرفت. این کاراکترها (که فهرست آنها در استاندارد HTML آمده) با & شروع و به ; ختم می‌شوند. برای مثال، توالی گریز   معادل یک فاصله (space)، توالی گریز è معادل è، و توالی گریز é معادل é است. از آنجائیکه < ، > و & در فایلهای HTML معنای خاصی دارند، برای نمایش آنها نیز باید از توالی‌های گریز (بترتیب < ، > و &) استفاده کرد.

مهمترین چیزی که در قسمت سر نوشته می‌شود عنوان صفحهٔ وب است، که بین برچسبهای <title> و </title> قرار می‌گیرد (البته برخی اطلاعات دیگر، که به **اطلاعات‌متا** معروفند، نیز در قسمت سر نوشته می‌شوند). هر چیزی که در برچسب <title> نوشته شود، در خود صفحهٔ وب دیده نخواهد شد، بلکه در میلهٔ عنوان پنجرهٔ مرورگر ظاهر می‌شود.

اجازه دهید نگاهی به قسمتهای دیگر شکل ۷-۲۶ بیندازیم. برچسبهایی که در این فایل بکار رفته (و چندبرچسب دیگر) را در شکل ۷-۲۷ ملاحظه می‌کنید. برای نوشتن تیتر مطالب می‌توانید از برچسب <h*n*> ، که *n* عددی بین 1 تا 6 است، استفاده کنید: <h1> بزرگترین تیتر، و <h6> کوچکترین تیتر را تولید می‌کنند. فرمت کردن تیترها (اندازه و نوع فونت یا رنگ آنها) بر عهدهٔ مرورگر است، و معمولاً تیترهای درشت‌تر اعداد کوچکتری دارند. برچسب <h1> که بزرگترین تیتر را تولید می‌کند، دارای بیشترین فاصلهٔ خالی در بالا و پائین نیز هست، در حالیکه تیترهای کوچکتر فاصلهٔ کمتری با خطوط دیگر دارند.

برچسب	مفهوم
<html> ... </html>	صفحه وب را تعریف می کند
<head> ... </head>	محتویات سرصفحه را مشخص می کند
<title> ... </title>	عنوان صفحه را تعیین می کند (که البته در خود صفحه دیده نمی شود)
<body> ... </body>	بدنه صفحه را مشخص می کند
<h *n*> ... </h*n*>	سطح (اندازه) عنوان را تیتر را مشخص می کند
 ... 	محتویات برچسب را ضخیم می کند
<i> ... </i>	محتویات برچسب را کج می کند
<center> ... </center>	محتویات برچسب را واسط صفحه وب قرار می دهد
...	یک لیست غیر منظم (گلوله دار) می سازد
...	یک لیست شماره دار می سازد
...	آیتمهای لیست را مشخص می کند
 	یک شکست خط در صفحه ایجاد می کند
<p>	یک پاراگراف جدید می سازد
<hr>	یک خط افقی روی صفحه رسم می کند
<img src="...">	یک تصویر روی صفحه نمایش می دهد
 ... 	یک لینک را تعریف می کند

شکل ۷-۲۷. چند برچسب HTML . برخی از این برچسبها می‌توانند پارامتر داشته باشند.

برچسبهای <b> و <i> بترتیب برای ضخیم و کج کردن فونت بکار می‌روند. اگر مرورگر نتواند فونتهای ضخیم یا کج را نمایش دهد، معمولاً آنها بگونه‌ای دیگر (رنگ متفاوت یا نگاتیو کردن حروف) از سایر قسمتها متمایز خواهد کرد.

در HTML مکانیزمها مختلفی برای ایجاد لیست (از جمله لیستهای تودرتو) وجود دارد. لیست‌ها با برچسب <ul> یا <ol> شروع می‌شوند، که در هر دو مورد برای مشخص کردن آیتمهای لیست از برچسب <li> استفاده می‌کنیم. برچسب <ul> یک **لیست‌مرتب‌نشده** (unordered list) می‌سازد، که آیتمهای آن با **گلوله** (bullet - ●) مشخص می‌شوند. برچسب <ol> یک **لیست‌مرتب‌شده** (ordered list) می‌سازد، که آیتمهای آن توسط مرورگر شماره‌گذاری خواهند شد.

برچسبهای
 ، <p> و <hr> بین قسمتهای مختلف متن فاصله می‌اندازند. برای فرمت کردن دقیق متن می‌توان از شیوه‌نامه (style sheet) نیز استفاده کرد. برچسب
 باعث می‌شود تا ادامهٔ متن از سر خط شروع شود (معمولاً مرورگرها بعد از
 خط خالی اضافه نمی‌کنند). برچسب <p> پاراگراف جدیدی را شروع می‌کند و بین آنها یک خط فاصله می‌اندازد (و حتی ممکنست **تورفتگی** - indent - ابتدای پاراگراف را هم رعایت کند). از نظر تئوری یک پاراگراف باید با <p> شروع و به </p> ختم شود، ولی برچسب </p> بندرت بکار برده می‌شود، و حتی اغلب آنهایی که صفحات وب می‌نویسند از وجود آن خبر ندارد. برچسب <hr> نیز یک خط افقی روی صفحه رسم می‌کند.

صفحات وب می‌توانند تصویر نیز داشته باشند. یک برچسب <img> تصویر مشخص شده را در همان نقطه‌ای که این برچسب نوشته شده، نمایش می‌دهد. برچسب <img> می‌تواند پارامترهای متعددی بگیرد. پارامتر *src* همان URL تصویر موردنظر است. فرمت این تصاویر جزیی از استاندارد HTML نیست، بلکه جزء قابلیتهای مرورگر محسوب می‌شود. اغلب مرورگرها می‌توانند تصاویر GIF و JPEG را نمایش دهند. مرورگرها می‌توانند از فرمتهای دیگر نیز پشتیبانی کنند، ولی این یک شمشیر دو لبه است: اگر عادت کنید از فرمتهای دیگر (مثلاً، BMP) در صفحات وب خود استفاده کنید، شاید بعضی از کاربران (که مرورگرهای متفاوتی دارند) نتوانند خلاقیت هنری شما را تحسین کنند!

برخی دیگر از پارامترهای <img> عبارتند از: *align* که تصویر را با خط کرسی متن تراز می‌کند (مقادیر *top* ، *middle* ، و *bottom* بترتیب معادل تراز بالا، وسط و پائین هستند)؛ *alt* که متن جایگزین تصویر را (وقتی کاربر تصاویر را غیرفعال کرده باشد) مشخص می‌کند؛ و *ismap* که نشان می‌دهد تصویر دارای نقاط فعال (قابل کلیک) است.

برای ایجاد اَبَرلینک در صفحات وب از برچسب <a>... استفاده می‌کنیم. برچسب <a> هم پارامترهای مختلفی دارد، از جمله *href* (URL مقصد لینک)، و *name* (نام اَبَرلینک). متن (یا تصویری) که بین <a> و قرار دارد، در صفحهٔ وب دیده خواهد شد، و کاربر با کلیک کردن این متن (یا تصویر) می‌تواند به صفحهٔ مقصد لینک برود. در زیر نمونه‌ای از یک اَبَرلینک را می‌بینید.

```
<a href="http://www.nasa.gov">NASA's home page</a>
```

که کاربر آنرا در مرورگر به این صورت خواهد دید:

<u>NASA's home page</u>

و اگر کاربر کنجکاو ما به ناسا علاقمند باشد و روی این لینک کلیک کند، مرورگر بلافاصله صفحهٔ *http://www.nasa.gov* را آورده و نمایش می‌دهد.

مثال زیر شکل دیگری از همان لینک بالاست:

```
<a href="http://www.nasa.gov"><img src="shuttle.gif" alt="NASA"></a>
```

در اینجا بجای عبارت <u>NASA's home page</u> از تصویر یک شاتل فضایی استفاده کرده‌ایم، و کاربر با کلیک

کردن این تصویر به همان صفحهٔ *http://www.nasa.gov* خواهد رفت. اگر هم کاربر نمایش تصاویر را در مرورگر غیرفعال کرده باشد، کلمهٔ NASA نشان می‌دهد که موضوع از چه قرار است.

برچسب <a> پارامتری دارد بنام *name*، که به کمک آن می‌توان لینک را به وسط یک صفحه نشانه رفت. با این پارامتر می‌توان صفحاتی بصورت فهرست مطالب ایجاد کرد، که کلیک کردن آیتمهای آن باعث رفتن کاربر به قسمت مربوطه در همان صفحه می‌شود.

استاندارد HTML از تغییر و تکامل در امان نبوده است. در ویرایشهای HTML 1.0 و HTML 2.0 جدول (table) وجود نداشت، ولی از HTML 3.0 این عنصر هم به صفحات وب اضافه شد. هر جدول HTML تعدادی سطر دارد، که خود از یک یا چند سلول (cell) تشکیل شده است. در یک سلول می‌توان هر چیزی قرار دارد: متن، عدد، تصویر، و حتی یک جدول دیگر. سلولها را می‌توان در هم ادغام کرد، برای مثال می‌توان با ادغام چند سلول برای جدول تیتر یا عنوان درست کرد. جدولها حرف آخر را در فرمت کردن صفحات وب (و کنترل خصوصیات ظاهری آن) می‌زنند.

در شکل ۷-۲۸ (الف) تعریف یک جدول HTML، و در شکل ۷-۲۸ (ب) خروجی آنرا می‌بینید. این مثال فقط برخی از ویژگیهای ابتدایی جدول را نشان می‌دهد. جدول‌ها با برچسب <table> شروع می‌شوند، و با دادن پارامترهای دلخواه می‌توان ویژگیهای کلی آنها را تعیین کرد.

با استفاده از برچسب <caption> می‌توان یک تیتر کلی به جدول داد. هر سطر جدول با برچسب <tr> (مخفف Table Row) شروع می‌شود، و برای مشخص کردن محتویات سلولها از برچسب‌های <th> (مخفف Table Header) یا <td> (مخفف Table Data) استفاده می‌شود (مرورگر برچسبهای <th> و <td> را بگونه‌ای متفاوت فرمت می‌کند). جدول‌ها دارای صفات متعددی دیگری (از جمله نوع تراز افقی و عمودی سلولها، حاشیهٔ آنها، و ادغام سلولها در یکدیگر) نیز هستند.

در HTML 4.0 قابلیتهای جدیدی به این زبان اضافه شده است، که از میان آنها می‌توان به امکانات جدید برای سهولت ویگردی معلولین، قابلیت استفاده از شیء (object) در صفحات وب، پشتیبانی از زبانهای اسکریپت‌نویسی (برای ایجاد محتویات دینامیک) اشاره کرد.

در سایتهای بزرگ وب که طراحان متعددی در ایجاد صفحات آن شرکت دارند، یکی از مهمترین نکات حفظ یکدستی و یکنواختی ظاهر صفحات است. این مشکل را می‌توان با استفاده از شیوه‌نامه (style sheet) حل کرد. در این تکنیک طراحان صفحات وب بجای شیوه‌های فیزیکی (مانند فونت ضخیم یا کج)، از شیوه‌های منطقی مانند <dn> (تعریف)، <em> (تأکید خفیف)، <strong> (تأکید شدید)، و <var> (متغیرهای برنامه) استفاده می‌کنند. این شیوه‌های منطقی در یک شیوه نامه (که در ابتدای هر صفحه به آن ارجاع می‌شود) تعریف می‌شوند. با استفاده از شیوه نامه، سرپرست تیم طراحی می‌تواند ظاهر یکنواخت کلیهٔ صفحات را تضمین کند. برای مثال، اگر روزی تصمیم گرفته شود که برچسب <strong> از فونت 14 کج آبی به فونت 18 ضخیم صورتی پُررنگ تبدیل شود، فقط کافیست تعریف این برچسب در شیوه نامه تغییر کند. شیوه نامه‌ها را می‌توان معادل فایلهای #include در برنامه‌های C (که محل تعریف ماکروهاست) دانست.

فرم

اولین ویرایش HTML (یعنی HTML 1.0) اساساً یک طرفه بود: کاربران می‌توانستند صفحات را درخواست کنند، ولی فرستادن اطلاعات به سرویس‌دهندهٔ وب بسیار مشکل بود. با رشد کاربردهای تجاری وب، تقاضا برای ترافیک دو طرفه بشدت فزونی گرفت. برای مثال، شرکتهای بسیاری میل داشتند از طریق وب هم سفارش بگیرند، یا شرکتهای نرم‌افزاری می‌خواستند محصولات خود را بصورت الکترونیکی بفروشند، و دهها مورد مانند آن.

```
<html>
<head> <title> A sample page with a table </title> </head>
<body>
<table border=1 rules=all>
<caption> Some Differences between HTML Versions </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th>Item  <th>HTML 1.0  <th>HTML 2.0  <th>HTML 3.0 <th>HTML 4.0 </tr>
<tr> <th> Hyperlinks <td> x <td> x <td> x <td> x </tr>
<tr> <th> Images <td> x <td> x <td> x <td> x </tr>
<tr> <th> Lists <td> x <td> x <td> x <td> x </tr>
<tr> <th> Active Maps and Images <td>   <td> x <td> x <td> x </tr>
<tr> <th> Forms <td>   <td> x <td> x <td> x </tr>
<tr> <th> Equations <td>   <td>   <td> x <td> x </tr>
<tr> <th> Toolbars <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tables <td>   <td>   <td> x <td> x </tr>
<tr> <th> Accessibility features <td>   <td>   <td>   <td> x </tr>
<tr> <th> Object embedding <td>   <td>   <td>   <td> x </tr>
<tr> <th> Scripting <td>   <td>   <td>   <td> x </tr>
</table>
</body>
</html>
```

(الف)

برخی از تفاوتهای در ویرایش HTML

آیتم	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
ابر لینک	x	x	x	x
تصویر	x	x	x	x
لیست	x	x	x	x
تصویر یا نقش فعال		x	x	x
فرم		x	x	x
معادله			x	x
میله ابزار			x	x
جدول			x	x
ویژگی معلولین				x
قرار دادن شی ء در صفحه				x
اسکریپت نویسی				x

(ب)

شکل ۷–۲۸. (الف) یک جدول HTML . (ب) خروجی جدول.

این تقاضاها باعث شد تا فرم (form) به ویرایش بعدی یعنی HTML 2.0 اضافه شود. فرم می‌تواند فیلدها (یا دکمه‌هایی) داشته باشد، که کاربر آنها را پُر کرده و به سرویس‌دهنده برگرداند. برای این منظور از برچسبی بنام <input> استفاده می‌کنیم، که دارای پارامترهای متعددی برای کنترل ظاهر و عملکرد آن است. رایجترین فرمها معمولاً دارای یک یا چند فیلد برای وارد کردن متن، یک یا چند جعبه برای انتخاب کردن، نگاشتهای فعال، و دکمه‌های *submit* باشند. در شکل ۷-۲۹ برخی از خصوصیات فرم نشان داده شده است.

اجازه دهید برای آشنایی بیشتر با فرمها، این مثال را دقیقتر بررسی کنیم. هر فرم با برچسبهای <form> و </form> مشخص می‌شود. در داخل یک فرم می‌توان از تمام برچسبهای HTML استفاده کرد، اما هر متنی که خارج این برچسبها نوشته شود، بهمان صورت دیده خواهد شد. سه نوع **جعبهٔ‌ورودی** (input box) وجود دارد که می‌توان از آنها در فرمها استفاده کرد.

در شکل ۷-۲۹، اولین جعبهٔ ورودی بعد از کلمهٔ "Name" آمده است. این جعبه ورودی رشته‌ایی بطول حداکثر ۴۶ کاراکتر می‌گیرد، و آنرا در متغیری بنام *customer* ذخیره می‌کند. برچسب <p> هم باعث می‌شود که مرورگر فیلدهای بعدی را در خط بعد نشان دهد. با استفاده از برچسب <p> طراح صفحه می‌تواند ظاهر آنرا بصورت دلخواه کنترل کند.

خط بعدی فرم یک آدرس (بطول ۴۰ کاراکتر) از کاربر می‌گیرد. فیلدهای شهر، استان، و کشور هم بدنبال آن آمده‌اند؛ بین این فیلدها <p> وجود ندارد، بنابراین مرورگر آنها را در یک خط نشان می‌دهد. تا آنجاکه به مرورگر مربوط است، این پاراگراف شش آیتم دارد (سه رشتهٔ متن، و سه جعبهٔ ورودی)، که باید آنها را یک خط نمایش دهد (و اگر نتوانست به خط بعدی می‌رود). اگر وضوح تصویر مانیتور زیاد باشد، این شش آیتم در یک خط دیده خواهند شد، اما اگر وضوح آن پائین باشد، احتمالاً به دو خط شکسته می‌شوند (حتی ممکنست کلمهٔ "Country" در آخر خط، و جعبهٔ ورودی مقابل آن در ابتدای خط بعدی بیفتد).

خط بعدی شمارهٔ کارت اعتباری و تاریخ انقضای آنرا از کاربر می‌گیرد (البته ارسال این قبیل اطلاعات روی اینترنت فقط باید در شرایط کاملاً امن صورت گیرد - در فصل ۸ در این باره بیشتر صحبت خواهیم کرد).

بعد از فیلد تاریخ انقضا، نوع دیگری از جعبه ورودی را می‌بینید: **دکمهٔ‌رادیویی** (radio button). از دکمهٔ رادیویی وقتی استفاده می‌شود که بخواهیم انتخاب کاربر را به دو یا چند گزینه محدود کنیم (مانند رادیوی ماشین که با زدن هر دکمه می‌توان یک ایستگاه از پیش تعیین شده را انتخاب کرد). کاربر می‌تواند برای انتخاب هر یک از این دکمه‌ها روی آن کلیک کند (و یا از صفحه کلید استفاده کند). انتخاب هر دکمه، باعث می‌شود که دکمه‌های دیگر همان گروه خاموش شوند (نمایش ظاهری دکمه‌های خاموش و روشن بر عهدهٔ مرورگر است). گروه‌بندی دکمه‌های رادیویی (و ایجاد ارتباط منطقی بین آنها) توسط صفت *name* صورت می‌گیرد. برای مثال، Widget size گروه مستقلیست که آن هم دو دکمه دارد.

برای مشخص کردن اینکه در هر گروه کدام دکمه روشن است، از پارامتر *value* استفاده می‌کنیم. برای مثال، در گروه اول مقدار متغیر cc ("mastercard" یا "visacard") نشان می‌دهد که کاربر کدامیک از دکمه‌های M/C یا Visa را انتخاب کرده است.

بعد از این دکمه‌های رادیویی، نوع دیگری از جعبه ورودی را می‌بینید: **جعبه‌چک** (checkbox). یک جعبه چک نیز مانند دکمهٔ رادیویی می‌تواند دو حالت داشته باشد: روشن، خاموش. اما بر خلاف دکمه‌های رادیویی، هر جعبه چک می‌تواند مستقلاً خاموش یا روشن باشد. برای مثال، وقتی می‌خواهید از طریق وب سفارش پیتزا بدهید، می‌توانید قارچ، گوشت و پیاز را همزمان بعنوان ترکیبات یک پیتزا انتخاب کنید، ولی امکان انتخاب همزمان اندازه‌های کوچک، متوسط و بزرگ را برای یک پیتزا ندارید. بهمین دلیل ترکیبات پیتزا را با استفاده از جعبه چک نشان می‌دهند، و اندازهٔ آنرا با دکمه‌های رادیویی.

```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

(الف)

Widget Order Form

Name

Street address

City State Country

Credit card # Expires M/C Visa

Widget size Big Little Ship by express courier

Submit order

Thank you for ordering an AWI widget, the best widget money can buy!

(ب)

شکل ۷-۲۹. (الف) یک فرم HTML . (ب) خروجی فرمت شدهٔ فرم.

در مواردی که تعداد گزینه‌های موجود برای یک انتخاب بسیار زیاد باشد، استفاده از دکمه‌های رادیویی قدری مشکل است. در این موارد می‌توان از برچسب <select> استفاده کرد: این برچسب با پارامتر *multiple* مانند

جعبه چک عمل می‌کند، و بدون آن مانند دکمهٔ رادیویی. در اغلب مرورگرها برچسب <select> بصورت یک منوی بازشو (drop-down menu) نمایش داده می‌شود.

تا اینجا دو نوع <input> را دیدید: *radio* و *checkbox* . البته نوع سوم را هم قبل از آن دیده بودید: نوع *text* ؛ ولی از آنجائیکه این نوع پیش‌فرض <input> است، استفاده از صفت *type = text* ضرورتی ندارد. انواع دیگر جعبه ورودی عبارتند از: *password* و *textarea* . جعبهٔ *password* درست مانند *text* است، با این تفاوت که هر چیزی که در آن نوشته شود، فقط * نشان می‌دهد. جعبهٔ *textarea* نیز مانند *text* است، فقط چندخطی است.

در آخرین قسمت از فرم شکل ۷-۲۹ یک دکمهٔ *submit* می‌بینید. وقتی کاربر این دکمه را کلیک کند، اطلاعاتی که در سایر قسمتهای فرم وارد کرده به کامپیوتری که فرم روی آن قرار دارد، فرستاده می‌شود. در اینجا پارامتر *value* عبارتیست که روی دکمهٔ *submit* دیده خواهد شد. دکمهٔ *submit* تنها جعبه ورودی است که باید *value* داشته باشد، در سایر جعبه‌ها این پارامتر اختیاری است (چون کاربر می‌تواند متن موجود در آنها را بدلخواه تغییر دهد).

با کلیک شدن دکمهٔ *submit* ، مرورگر تمام اطلاعات فرم را در یک خط ترکیب کرده و به سرویس‌دهنده برمی‌گرداند. این فیلدها با & از هم جدا می‌شوند، و اگر در آنها فاصله وجود داشته باشد، مرورگر بجای آن + قرار می‌دهد. در شکل ۷-۳۰ نمونه‌ای از مقدار برگشتی فرم ۷-۲۹ به سرویس‌دهنده را می‌بینید (تمام این اطلاعات در واقع فقط یک خط است، که بدلیل محدودیت عرض صفحه به چند خط شکسته شده است).

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&
product=cheap&express=on
```

شکل ۷-۳۰. ورودیهای کاربر که مرورگر به سرویس‌دهنده برمی‌گرداند.

(اگر یک جعبه چک انتخاب نشده باشد، مرورگر آنرا به سرویس‌دهنده نمی‌فرستد.) تفسیر اطلاعات رسیده بر عهدهٔ سرویس‌دهنده است - در این باره بعداً بیشتر صحبت خواهیم کرد.

XML و XSL

HTML ، با فرم یا بدون آن، هیچگونه ساختاری برای صفحات وب فراهم نمی‌آورد. در HTML محتوای صفحه و فرمانهای فرمت‌کننده نیز مخلوط هستند. با گسترش روزافزون تجارت الکترونیک (و سایر کاربردها)، نیاز به نوعی ساختار برای صفحات وب (و تفکیک محتوا و فرمت) بالا گرفت. برای مثال، برنامه‌ای که می‌خواهد در وب بدنبال بهترین قیمت یک کتاب (یا CD) جستجو کند، باید صفحات متعددی را خوانده و در آنها بدنبال عنوان کتاب و قیمت آن بگردد. وقتی صفحات وب با HTML نوشته شده باشند، برای چنین برنامه‌ای دشوار است، تشخیص دهد عنوان کتاب کجاست و قیمت آن کجا.

بهمین دلیل کنسرسیوم W3C استاندارد جدیدی برای HTML توسعه داده، که به کمک آن می‌توان به صفحات وب ساختاری مناسب برای پردازش خودکار داد. برای این هدف دو زبان جدید نیز توسعه داده شده است. اولی، XML (زبان علامتگذاری قابل توسعه - eXtensible Markup Language)، محتویات صفحهٔ وب را بصورت ساخت‌یافته توصیف می‌کند، و دومی، XSL (زبان شیوه‌نویسی قابل توسعه - eXtensible Style Language)، برای توصیف مستقل فرمت این محتواست. هر دوی این زبانها بسیار مفصل و پیچیده‌اند، و بحث این قسمت فقط ایده‌ای از طرز کار آنها به شما خواهد داد.

مثال شکل ۷-۳۱ را در نظر بگیرید. در این مثال ساختاری بنام book_list ، برای نگهداری مشخصات کتابها، تعریف شده است. هر کتاب سه فیلد دارد: عنوان، مؤلف، و سال انتشار. این ساختار بسیار ساده است، اما می‌توان ساختارهای پیچیده‌تری هم ایجاد کرد (مانند کتابهایی که چند مؤلف دارند، کتابهایی که CD پیوست دارند، و یا URL سایتهای عرضه و فروش کتاب).

در این مثال هر فیلد فقط یک قسمت دارند، اما فیلدها را می‌توان به فیلدهای کوچکتری نیز تقسیم کرد. بعنوان مثال، می‌توان فیلد <author> را به نام و نام خانوادگی تقسیم کرد، تا جستجوهای دقیقتر ممکن شود (این تقسیم را می‌توان تا هر درجه‌ای از عمق انجام داد):

```
<author>
    <first_name>Andrew</first_name>
    <last_name>Tanenbaum</last_name>
</author>
```

تمام کاری که فایل ۷-۳۱ انجام می‌دهد، ایجاد فهرستی از سه کتاب است. این فایل هیچ حرفی دربارهٔ نحوهٔ نمایش این اطلاعات نمی‌زند. برای فرمت کردن این اطلاعات به فایل دیگری نیاز داریم: فایل *book_list.xsl* ، که حاوی تعریفهای XSL لازم برای فرمت کردن فهرست *book_list.xml* است. این فایل یک شیوه‌نامه است که نحوهٔ فرمت کردن صفحهٔ وب را بیان می‌کند. (البته روشهای دیگری برای فرمت کردن فایلهای XML وجود دارد - مانند تبدیل XML به HTML - که از حوزهٔ بحث این کتاب خارج است.)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="b5.xsl"?>
<book_list>
<book>
   <title> Computer Networks, 4/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 2003 </year>
</book>
<book>
   <title> Modern Operating Systems, 2/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 2001 </year>
</book>
<book>
   <title> Structured Computer Organization, 4/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 1999 </year>
</book>
</book_list>
```

شکل ۷-۳۱. یک صفحهٔ وب که با XML نوشته شده است.

در شکل ۷-۳۲ یک فایل XSL نمونه برای فرمت کردن فایل ۷-۳۱ را ملاحظه می‌کنید. بعد از چند تعریف لازم (مانند URL استاندارد XSL)، برچسبهای <html> و <body> آمده‌اند، که اینها (مانند همیشه) شروع صفحهٔ وب را مشخص می‌کنند. پس از آن جدولی با یک تیتر و سه ستون تعریف کرده‌ایم. دقت کنید که در این

جدول علاوه بر برچسبهای <th> از برچسبهای </th> نیز استفاده کرده‌ایم، کاری که قبلاً نمی‌کردیم. استانداردهای XML و XSL بسیار سختگیرتر از HTML هستند، و رعایت اصول نوشتن برچسبها در آنها لازم است، حتی اگر مرورگر قادر به تشخیص نیت طراح صفحهٔ وب باشد. مرورگری که فایلهای غلط XML و XSL را قبول کند (و خود اشکالات آنها را رفع کند)، در امتحانات سازگاری نمرهٔ قبولی نخواهد گرفت - مرورگرها فقط می‌توانند خطاها را اعلام کنند. این مقررات سختگیرانه برای مقابله با افراد تنبلی که وب را پُر از صفحات شلخته کرده‌اند، لازم است.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<html>
<body>
<table border="2">
  <tr>
     <th> Title</th>
     <th> Author</th>
     <th> Year </th>
  </tr>

  <xsl:for-each select="book_list/book">
  <tr>
     <td> <xsl:value-of select="title"/> </td>
     <td> <xsl:value-of select="author"/> </td>
     <td> <xsl:value-of select="year"/> </td>
  </tr>
  </xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

شکل ۷-۳۲. یک شیوه‌نامهٔ XSL .

دستور

```
<xsl:for-each select="book_list/book">
```

معادل دستور for در زبان C است. این دستور (که به </xsl:for-each> ختم می‌شود) در یک حلقه کتابها را بترتیب می‌خواند. در هر تکرار این حلقه پنج فرمان HTML در خروجی نوشته می‌شود: <tr> ، عنوان کتاب، مؤلف، سال انتشار کتاب، و </tr> . پس از اتمام حلقه و بستن جدول، برچسبهای </body> و </html> نیز در خروجی نوشته می‌شوند. حاصل کار یک جدول HTML است، که کاملاً شبیه جدولهای معمولی است. اما با این‌فرمت هر برنامه‌ای می‌تواند فایل XML را آنالیز کرده، و کتابهای موردنظر را استخراج کند. تأکید بر این نکته ضروریست که، اگر چه در فایل XSL نوعی حلقه وجود دارد ولی صفحات XML و XSL همچنان استاتیک هستند، چون فقط برای فرمت کردن صفحهٔ وب بکار می‌آیند. البته مرورگر باید توانایی تفسیر فایلهای XML و XSL را داشته باشد، ولی خبر خوب اینست که اغلب مرورگرهای امروزی چنین قابلیتی دارند. آیندهٔ XSL و اینکه آیا می‌تواند جایگزین شیوه‌نامه‌ها شود، هنوز در پرده‌ای از ابهام قرار دارد.

با اینکه روش کار را نشان ندادیم، XML به طراح سایت وب اجازه می‌دهد تا با تعریف فایلهای ساختاری و ضمیمه کردن آنها به صفحات وب، سایتهایی پیچیده‌ای ایجاد کند. کتابهای متنوع و بسیار خوبی در این زمینه نوشته شده، که برای نمونه می‌توان به (Livingston, 2002; and Williamson, 2001) اشاره کرد.

قبل از پایان دادن به بحث XML و XSL ، بد نیست به جنگ ایدئولوژیکی که بین کنسرسیوم WWW و طراحان وب در گرفته، هم اشاره کنیم. هدف اولیهٔ HTML مشخص کردن ساختار سند بود، نه ظاهر آن. برای مثال، فرمان

```
<h1>Deborah's Photos</h1>
```

فقط به مرورگر می‌گوید که این یک تیتر است، اما هیچ چیز دربارهٔ فرمت (نوع فونت، رنگ و یا اندازهٔ) آن نمی‌گوید. این وظیفه بر عهدهٔ مرورگر گذاشته شده است. اما از آنجائیکه بسیاری از طراحان وب مایلند ظاهر صفحه را خود کنترل کنند، برای این کار برچسبهای جدیدی به HTML اضافه شد:

```
<font face="helvetica" size="24" color="red">Deborah's Photos</font>
```

همچنین فرمانهایی برای کنترل دقیق محل اشیاء روی صفحه در نظر گرفته شد. اما مشکل این روش آن است که صفحاتی که با این دستورات نوشته می‌شوند، همه جا یکسان دیده نمی‌شوند (صفحه‌ای که در کامپیوتر طراح آن بسیار قشنگ دیده می‌شود، ممکنست در جای دیگر یک افتضاح واقعی باشد). XML تلاشی بود برای بازگشت به آن ایده‌های اولیه: مشخص کردن ساختار صفحه، نه ظاهر آن. اما هر ایدهٔ خوبی می‌تواند مورد سوءاستفاده قرار گیرد (در این حرف شک نکنید!).

علاوه بر توصیف صفحات وب ، XML کاربردهای دیگری نیز دارد. برای مثال، می‌توان از XML بعنوان رابط بین برنامه‌های کاربردی استفاده کرد: SOAP (**پروتکل سادهٔ دسترسی شیء** – Simple Object Access Protocol) یکی از راههای انجام RPC (**فراخوانی از راه دور** – Remote Procedure Call) بین برنامه‌ها (مستقل از زبان و سیستم عامل) است. در این روش، مشتری درخواست خود را بصورت یک پیام XML در آورده و با استفاده از پروتکل HTTP به سرویس‌دهنده می‌فرستد؛ پاسخ سرویس‌دهنده هم بصورت پیام XML به مشتری برگشت داده می‌شود. با این تکنیک برنامه‌هایی که روی سیستم عامل‌های متفاوت هستند، می‌توانند با یکدیگر ارتباط برقرار کنند.

XHTML

HTML برای پاسخ به نیازهای جدید به تحول خود ادامه می‌دهد. از هم اکنون می‌توان حدس زد که در آینده صفحات وب فقط به PC ها محدود نمی‌مانند، و راه خود را به دستگاههای تلفن همراه و PDA ها باز خواهند کرد. این قبیل دستگاهها حافظهٔ کمی دارند، و نمی‌توانند مرورگرهای حجیم و سنگین امروزی (که قسمت اعظم کُد آنها صرف حدس زدن و رفع و رجوع خرابکاری طراح صفحهٔ وب می‌شود) را اجرا کنند. بهمین دلیل، بعد از HTML 4 (بجای HTML 5) زبان جدیدی بنام XHTML (HTML **توسعه‌یافته** – eXtended HTML) به بازار آمد، که بسیار هم ناخن‌خشک است. این زبان جدید اساساً همان HTML 4 است، که با XML فرمول‌بندی شده است. بعبارت دیگر، در این زبان برچسبی مانند <h1> هیچ معنای خاصی ندارد، مگر اینکه در یک فایل XSL تعریف شده باشد. XHTML استاندارد جدید وب است، و اگر می‌خواهید بالاترین قابلیت جابجایی را در وب داشته باشید، باید از آن استفاده کنید.

بین HTML 4 و XHTML شش تفاوت عمده (و چند تفاوت جزئی) وجود دارد، که در اینجا آنها را فهرست‌وار مرور خواهیم کرد. اول اینکه، صفحات و مرورگرهای XHTML باید استانداردهای آن را دقیقاً رعایت کنند (دیگر جایی برای بنجل‌ها نیست!). XHMTL این ویژگی را از XML ارث برده است.

دوم، تمام برچسب‌ها و صفت‌های XHTML باید با حروف کوچک نوشته شوند. برچسبی مثل <HTML> دیگر در XHTML معتبر نیست، و باید آنرا بصورت <html> نوشت. دستور <img SRC="pic001.jpg"> نیز غلط است، چون XHTML صفتی بنام SRC نمی‌شناسد.

سوم، حذف برچسبهای انتهایی (حتی برچسبی مثل </p>) قدغن است. در برچسبهایی که ذاتاً انتها ندارند (مانند
 ، <hr> ، و <img>)، نیز باید از /> استفاده کرد:

```
<img src="pic001.jpg" />
```

چهارم، مقدار تمام صفت‌ها باید در داخل گیومه نوشته شود (حتی اگر عدد باشد). مثلاً، دستور

```
<img src="pic001.jpg" height=500 />
```

مجاز نیست، چون 500 در داخل "" قرار ندارد.

پنجم، تودرتو کردن برچسبها باید بدرستی انجام شود. در گذشته این کار لزومی نداشت، چون مرورگر منظور طراح صفحه را حدس می‌زد. برای مثال، HTML 4 دستور زیر را بدرستی اجرا می‌کند:

```
<center><b>Vacation Pictures</center></b>
```

اما این دستور در XHTML مجاز نیست (جای </center> و باید عوض شود).

و بالاخره ششم اینکه، نوع هر سند باید دقیقاً (در ابتدای آن) مشخص شده باشد. (برای دیدن سایر تفاوتهای XHTML و HTML می‌توانید به سایت *www.w3c.org* مراجعه کنید.)

۷-۳-۳ سندهای وب دینامیک

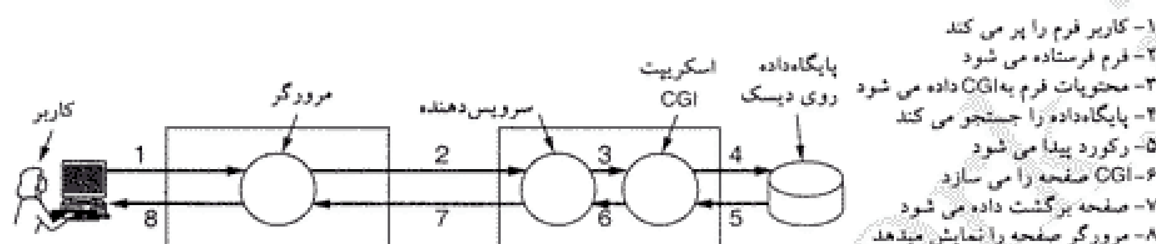
تا اینجا با مدلی که در شکل ۶-۶ دیدید سروکار داشتیم: مشتری فایلی را از سرویس‌دهنده درخواست می‌کند، و سرویس‌دهنده این فایل را برای او می‌فرستد. در روزهای اولیهٔ وب اوضاع همیشه بر همین منوال بود، یعنی صفحه‌ها ثابت بودند و هیچ تغییری نمی‌کردند. اما در سالهای اخیر روز به روز بر تعداد صفحاتی که محتویات آنها بصورت دینامیک تولید می‌شود (و از قبل فایلی روی دیسک سرویس‌دهنده وجود ندارد)، اضافه شده است. تولید محتوا می‌تواند در دو نقطه صورت گیرد: سمت سرویس‌دهنده، و سمت مشتری. اجازه دهید هر یک از آنها را بررسی کنیم.

تولید صفحات وب دینامیک سمت-سرویس‌دهنده

برای اینکه ببینید اساساً چرا تولید محتوا در سمت سرویس‌دهنده احتیاج است، فرمهایی را که در قسمت قبل دربارهٔ آنها صحبت کردیم، در نظر بگیرید. وقتی کاربر فرم را پُر کرده و دکمهٔ *submit* را کلیک می‌کند، پیامی به سرویس‌دهنده فرستاده شده و مقدار فیلدهای فرم را به آن اعلام می‌کند. این پیام هیچ چیز دربارهٔ اینکه سرویس‌دهنده چه فایلی باید برگرداند، نمی‌گوید – و در واقع سرویس‌دهنده قبل از هر کاری باید این اطلاعات را (توسط یک برنامه یا اسکریپت) پردازش کند. معمولاً از اطلاعات فرم برای جستجوی یک پایگاه داده (روی سرویس‌دهنده) و ایجاد یک صفحهٔ HTML خاص استفاده می‌شود. برای مثال، وقتی در یک برنامهٔ تجارت الکترونیک کاربر دکمهٔ *PROCEED TO CHECKOUT* را کلیک می‌کند، مرورگر کوکی محتوی اقلام موجود در سبد خرید کاربر را هم به همراه آن به سرویس‌دهنده می‌فرستد، و این سرویس‌دهنده است که باید با استفاده از اطلاعات این کوکی صفحهٔ HTML لازم را ایجاد کند. بعنوان مثال، این صفحه می‌تواند اقلام انتخاب شده را به کاربر نشان دهد، و ضمن نمایش اطلاعات دیگر (از قبیل آدرس خریدار، و شمارهٔ کارت اعتباری) تأئید نهایی وی را برای شروع عملیات مالی اخذ کند. در شکل ۷-۳۳ مراحل پردازش اطلاعات فرم HTML نشان داده شده است.

روش سنتی پردازش فرمها و دیگر صفحات تعاملی وب سیستمی است بنام CGI (**واسط مشترک دروازه** –

Common Gateway Interface). CGI یک واسط استاندارد شده است که به سرویس‌دهندهٔ وب اجازه می‌دهد تا با برنامه‌ها یا اسکریپت‌های پشت صحنه (که می‌توانند اطلاعات ورودی را از فرمها گرفته، و صفحات HTML تولید کنند) ارتباط برقرار کند. برنامه‌های پشت صحنه معمولاً به زبان اسکریپت‌نویسی پرل (Perl) نوشته می‌شوند، چون نوشتن آنها ساده‌تر و سریعتر است (البته اگر پرل بلد باشید). این قبیل برنامه‌ها را معمولاً در یک دایرکتوری بنام *cgi-bin* ذخیره می‌کنند (که در اغلب URL ها می‌توانید این موضوع را ببینید). زبانهای اسکریپت‌نویسی دیگری هم هست (مانند پایتون – Python)، که می‌توان بجای پرل از آنها استفاده کرد.



شکل ۷-۳۳. مراحل پردازش اطلاعات یک فرم HTML .

برای دیدن طرز کار CGI ، فرض کنید شرکت Truly Great Products Company محصولات خود را بدون کارت ضمانت‌نامه می‌فروشد، ولی از خریداران دعوت می‌کند که برای پُر کردن کارت ضمانت‌نامه به سایت *www.tgpc.com* مراجعه کنند. وقتی خریدار به این صفحه مراجعه می‌کند، لینک زیر را در آن می‌بیند:

Click here to register your product

این لینک به اسکریپت *www.tgpc.com/cgi-bin/reg.perl* اشاره می‌کند. وقتی این اسکریپت بدون هیچ پارامتری اجرا شود، یک صفحهٔ HTML شامل فرم ضمانت‌نامه به کاربر برمی‌گرداند. وقتی کاربر بعد از پُر کردن فرم دکمهٔ *submit* را کلیک کند، پیامی محتوی اطلاعات فرم به اسکریپت *reg.perl* فرستاده می‌شود. اسکریپت *reg.perl* پس از پردازش اطلاعات فرم، یک رکورد برای مشتری جدید در پایگاه دادهٔ مشتریان ایجاد کرده، و یک صفحهٔ HTML حاوی شمارهٔ ضمانت‌نامه محصول و تلفن پشتیبانی پس از فروش را به کاربر برمی‌گرداند. این تنها راه پردازش دینامیک اطلاعات نیست، اما متداولترین روش است. صدها کتاب دربارهٔ اسکریپت‌های CGI و برنامه‌نویسی پرل نوشته شده، که بعنوان نمونه می‌توانیم به (Hanegan, 2001; Lash, 2002; and Meltzer and Michalski, 2001) اشاره کنیم.

اسکریپت‌های CGI تنها روش تولید محتویات دینامیک در سمت سرویس‌دهنده نیست. روش متداول دیگر نوشتن اسکریپت‌های کوچک در داخل صفحات HTML و اجرای این اسکریپت‌ها توسط سرویس‌دهنده است. یکی از زبانهای رایج برای نوشتن این قبیل اسکریپت‌ها PHP (**صفحهٔ خانگی شخصی** – Personal Home Page) نام دارد. البته سرویس‌دهنده باید این زبان را بشناسد تا بتواند اسکریپت‌های آنرا اجرا کند (درست مثل مرورگر که باید XML را بشناسد تا بتواند صفحات XML را نمایش دهد). صفحاتی که حاوی کُد PHP هستند، معمولاً بجای *htm* یا *html* پسوند *php* دارند.

در شکل ۷-۳۴ یک اسکریپت کوچک PHP را ملاحظه می‌کنید؛ این اسکریپت با هر سرویس‌دهنده‌ای که PHP روی آن نصب شده باشد، کار می‌کند. در این صفحه (علاوه بر برچسبهای معمول HTML) یک دستور PHP که در داخل برچسب <?php ... ?> قرار دارد، می‌بینید. این صفحه به کاربر اعلام می‌کند که از وی چه می‌داند. این قبیل اطلاعات معمولاً توسط مرورگر (بهمراه کوکی‌ها) به سرویس‌دهنده فرستاده می‌شوند (و

سرویس‌دهنده آنها را از طریق متغیر *HTTP_USER_AGENT* بدست می‌آورد). اگر این صفحه در فایلی بنام *test.php* (در سایت abcd.com) قرار داشته باشد، و کاربر روی لینک *www.abcd.com/test.php* کلیک کند، صفحه‌ای دریافت می‌کند که نوع مرورگر، زبان و سیستم عاملش را به وی اعلام خواهد کرد.

```
<html>
<body>
<h2> This is what I know about you </h2>
<?php echo $HTTP_USER_AGENT ?>
</body>
</html>
```

شکل ۷-۳۴. یک صفحهٔ HTML با دستورات PHP .

PHP از CGI ساده‌تر، و بویژه برای پردازش فرمها مناسبتر است. برای آشنایی بیشتر با طرز کار PHP مثال شکل ۷-۳۵(الف) را در نظر بگیرید. در این شکل یک فرم HTML می‌بینید؛ تنها تفاوت این فرم با فرمهای قبلی در خط اول آن است: این فرم هنگام کلیک شدن دکمهٔ *submit* فایلی بنام *action.php* را اجرا می‌کند. در این فرم دو جعبهٔ ورودی از نوع *text* وجود دارد، که یکی نام و دیگری سن کاربر را می‌پرسد. بعد از کلیک شدن دکمهٔ *submit* و ارسال محتویات فرم به سرویس‌دهنده (مانند آنچه در شکل ۷-۳۰ دیدید)، سرویس‌دهنده مقدار فیلد اول را در متغیری بنام *name* و مقدار فیلد دوم را در متغیری بنام *age* قرار می‌دهد. پس از آن فایل *action.php* (شکل ۷-۳۵ب) را پردازش کرده، و دستورات PHP آن را اجرا می‌کند. اگر کاربر در فیلدهای فرم ۷-۳۵(الف) بترتیب "Barbara" و "24" رابعنوان نام و سن خود وارد کرده باشد، فایلی شبیه شکل ۷-۳۵(ج) به وی برگردانده خواهد شد. همانطور که می‌بینید، پردازش فرمهای HTML با PHP بسیار ساده شده است.

با اینکه PHP نسبتاً آسان است، اما زبانی بسیار قوی برای ارتباط با پایگاه داده محسوب می‌شود. متغیرها، رشته‌ها، آرایه‌ها، و اغلب ساختارهای کنترلی PHP شباهت زیادی با C دارد (و I/O آن نیز بسیار قوی است). کُد PHP باز است و همه جا می‌توان آنرا مجانی بدست آورد. این زبان بویژه برای کار روی آپاچی (Apache - یکی از پُرطرفدارترین سرویس‌دهنده‌های وب) طراحی شده است. برای اطلاعات بیشتر دربارهٔ PHP به (Valade, 2002) مراجعه کنید.

تا اینجا دو روش مختلف برای ایجاد صفحات HTML دینامیک را بررسی کردیم: CGI و PHP . تکنیک سومی نیز وجود دارد، که JSP (**صفحات سرویس دهندهٔ جاوا** - JavaServer Pages) نام دارد؛ این تکنیک مشابه PHP است، با این تفاوت که بجای PHP از زبان برنامه‌نویسی **جاوا** (Java) استفاده می‌کند (صفحاتی که با این روش نوشته شده‌اند، پسوند *jsp* دارند). تکنیک چهارم یعنی ASP (**صفحات فعال سرویس دهنده** - Active Server Pages) معادلیست برای PHP و JSP از شرکت میکروسافت. در این تکنیک از زبان برنامه‌نویسی VBScript (که آن هم از محصولات میکروسافت است) استفاده می‌شود (صفحات ASP پسوند *asp* دارند). انتخاب یکی از تکنیکهای PHP ، JSP یا ASP بیشتر از آن که جنبهٔ فنی داشته باشد، سیاسی است (دعوای همیشگی طرفداران کُد باز، سان و میکروسافت)، چون همهٔ آنها تقریباً شبیه هم هستند.

به مجموعه تکنیکهای ایجاد محتویات دینامیک وب گاهی **HTML دینامیک** نیز گفته می‌شود.

تولید صفحات وب دینامیک سمت-مشتری

تکنیکهای CGI ، PHP ، JSP ، و ASP همگی برای پردازش اطلاعات فرمها (و ارتباط با پایگاه داده) در سمت

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(الف)

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

(ب)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 25
</body>
</html>
```

(د)

شکل ۷-۳۵. (الف) یک فرم HTML . (ب) اسکریپت PHP برای پردازش این فرم.
(ج) خروجی اسکریپت PHP برای ورودی‌های "Barbara" و "24" .

سرویس‌دهنده هستند. این تکنیک‌ها با پردازش اطلاعات فرم و جستجو در پایگاه داده، صفحات HTML مناسب را تولید و به مشتری برمی‌گردانند. اما هیچکدام از آنها نمی‌توانند حرکات ماوس را تشخیص داده، و یا مستقیماً با کاربر ارتباط برقرار کنند. برای این کار باید از اسکریپتهایی که در دل صفحات HTML نوشته شده و در کامپیوتر مشتری اجرا می‌شوند، استفاده کرد. پای این قبیل اسکریپتها، که با برچسب <script> مشخص می‌شوند، از HTML 4.0 به وب باز شد. متداولترین زبان اسکریپت‌نویسی سمت مشتری **جاوااسکریپت** (JavaScript) است، پس ما هم همین زبان را مختصراً بررسی خواهیم کرد.

جاوااسکریپت یک زبان اسکریپت‌نویسی است، که ارتباط دوری با جاوا دارد، اما مسلماً جاوا نیست. مانند سایر زبانهای اسکریپت‌نویسی، جاوااسکریپت زبانی سطح-بالا است. برای مثال، فقط با یک خط کُد می‌توان یک جعبه دیالوگ نمایش داد، ورودی کاربر را گرفت، و آنرا در یک متغیر ذخیره کرد. این ویژگی جاوااسکریپت را برای ایجاد صفحات وب **تعاملی** (interactive) بسیار مناسب کرده است. از طرف دیگر، این واقعیت که این زبان هنوز

استاندارد نشده (و با سرعتی بیشتر از یک مگس گرفتار آمده در ماشین اشعهٔ X جهش پیدا می‌کند)، نوشتن یک برنامهٔ جاوااسکریپت که بتواند روی هر سیستمی اجرا شود را فوق‌العاده مشکل کرده است.

برنامهٔ جاوااسکریپت شکل ۷-۳۶ را در نظر بگیرید (این برنامه هم مانند شکل ۷-۳۵(الف) نام و سن کاربر را گرفته، و پیش‌بینی می‌کند که وی سال آینده چند سال خواهد داشت!). قسمت <body> تقریباً با برنامهٔ PHP یکسان است؛ تنها جایی که تفاوت کرده، قسمت تعریف دکمهٔ *submit* است. در اینجا، صفت *onclick* به مرورگر می‌گوید که هنگام کلیک شدن دکمه باید اسکریپت *response* را اجرا کرده، و فرم *form* را بعنوان پارامتر به آن بدهد.

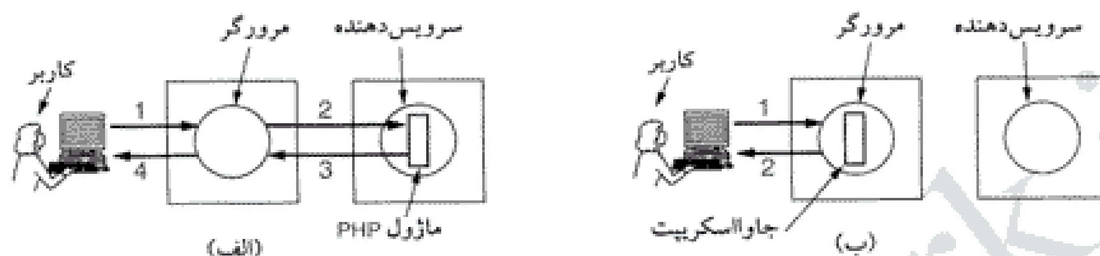
اما تعریف تابع جاوااسکریپت *response* در قسمت <head> این صفحه کاملاً جدید است. این تابع ابتدا مقدار فیلد *name* فرم را استخراج کرده و آنرا به همان صورت در متغیری بنام *person* ذخیره می‌کند؛ سپس مقدار فیلد *age* فرم را هم خوانده، و پس از تبدیل آن به عدد (با تابع *eval*) و اضافه کردن 1 به آن، آنرا در متغیر *years* ذخیره می‌کند. پس از آن یک سند (صفحهٔ وب) برای خروجی باز کرده، با دستور *writeln* چهار خط در آن می‌نویسد، و آنرا می‌بندد. بعد از کامل شدن این صفحه، مرورگر آنرا مانند یک فایل HTML معمولی نمایش می‌دهد.

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   var person = test_form.name.value;
   var years = eval(test_form.age.value) + 1;
   document.open();
   document.writeln("<html> <body>");
   document.writeln("Hello " + person + ".<br>");
   document.writeln("Prediction: next year you will be " + years + ".");
   document.writeln("</body> </html>");
   document.close();
}
</script>
</head>
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

شکل ۷-۳۶. استفاده از جاوااسکریپت برای پردازش فرم.

درک این نکته بسیار مهم است، که با وجود شباهت برنامه‌های ۷-۳۵ و ۷-۳۶، آنها بطریق کاملاً متفاوتی پردازش می‌شوند. در شکل ۷-۳۵، بعد از آن که کاربر دکمهٔ *submit* را کلیک کرد، مرورگر اطلاعات فرم را در رشته‌ای مانند شکل ۷-۳۰ جمع‌آوری کرده، و به سرویس‌دهنده‌ای که صفحه روی آن قرار دارد، می‌فرستد. سرویس‌دهنده بعد از دیدن نام فایل PHP ، این اسکریپت را اجرا می‌کند. اسکریپت PHP مزبور نیز با استفاده از اطلاعات فرم، یک صفحهٔ HTML ایجاد کرده و به مشتری برمی‌گرداند. اما وقتی در شکل ۷-۳۶ دکمهٔ *submit*

کلیک می‌شود، مرورگر اسکریپت درون آنرا اجرا می‌کند - تمام کارها در داخل مرورگر انجام می‌شود، و هیچ تماسی با سرویس‌دهنده وجود ندارد. بهمین دلیل نتیجهٔ کار معمولاً بلافاصله است، برخلاف اسکریپت PHP که رفت و برگشت صفحه ممکنست چندین ثانیه طول بکشد. تفاوت اسکریپت سمت-سرویس‌دهنده و سمت-مشتری در شکل ۷-۳۷ نشان داده شده است. در هر دو شکل، مرحلهٔ 1 گرفتن اطلاعات از کاربر است؛ تفاوت این دو روش در شکل بخوبی مشخص است.



شکل ۷-۳۷. (الف) اسکریپت سمت-سرویس‌دهنده با PHP . (ب) اسکریپت سمت-مشتری با جاوااسکریپت.

اما این تفاوت بدان معنا نیست که جاوااسکریپت بهتر از PHP است - آنها کاربردهای کاملاً متفاوتی دارند. PHP (و زبانهای مشابه آن) اساساً برای پردازش اطلاعات پایگاه داده روی وب مناسب هستند، در حالیکه جاوااسکریپت بدرد نوشتن برنامه‌های تعاملی می‌خورد. نوشتن صفحه‌ای که PHP و جاوااسکریپت را با هم داشته باشد، کاملاً ممکن است چون آنها کارهای متفاوتی انجام می‌دهند.

جاوااسکریپت یک زبان کامل است با تمام توانائیهای C و جاوا، و علاوه بر آن امکانات خاصی نیز برای پردازش صفحات وب (مانند کار با پنجره‌ها و فریمها، ست کردن و خواندن کوکی، پردازش فرم و لینک) دارد. در شکل ۷-۳۸ یک برنامهٔ جاوااسکریپت می‌بینید که در آن از یک تابع **بازگشتی** (recursive) استفاده شده است.

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   function factorial(n) {if (n == 0) return 1; else return n * factorial(n – 1);}
   var r = eval(test_form.number.value);      //  r = typed in argument
   document.myform.mytext.value = "Here are the results.\n";
   for (var i = 1; i <= r; i++)     // print one line from 1 to r
      document.myform.mytext.value += (i + "! = " + factorial(i) + "\n");
}
</script>
</head>
<body>
<form name="myform">
Please enter a number: <input type="text" name="number">
<input type="button" value="compute table of factorials" onclick="response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>
```

شکل ۷-۳۸. یک برنامهٔ جاوااسکریپت برای محاسبه و چاپ فاکتوریل.

جاوااسکریپت همچنین می‌تواند حرکت ماوس روی اشیاء مختلف صفحه را دنبال کند. بدین ترتیب می‌توان کاری کرد که وقتی کاربر ماوس را روی قسمتهای خاصی از صفحه می‌برد، اتفاق خاصی بیفتد (مثلاً، باز شدن یک منو، عوض شدن تصویر یا رنگ نوشته‌ها). این قبیل برنامه‌ها (که نمونه‌ای از آنرا در شکل ۷-۳۹می‌بینید) شادابی و سرزندگی خاصی به صفحات وب می‌دهند.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/ ast/im/kitten.jpg";
document.myurl[1] = "http://www.cs.vu.nl/ ast/im/puppy.jpg";
document.myurl[2] = "http://www.cs.vu.nl/ ast/im/bunny.jpg";
function pop(m) {
   var urx = "http://www.cs.vu.nl/ ast/im/cat.jpg";
   popupwin = window.open(document.myurl[m],"mywind","width=250,height=250");
}
</script>
</head>
<body>
<p> <a href="#" onMouseover="pop(0); return false;" > Kitten  </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;" > Puppy  </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;" > Bunny  </a> </p>
</body>
</html>
```

شکل ۷-۳۹. یک صفحهٔ وب تعاملی، که به حرکات ماوس عکس‌العمل نشان می‌دهد.

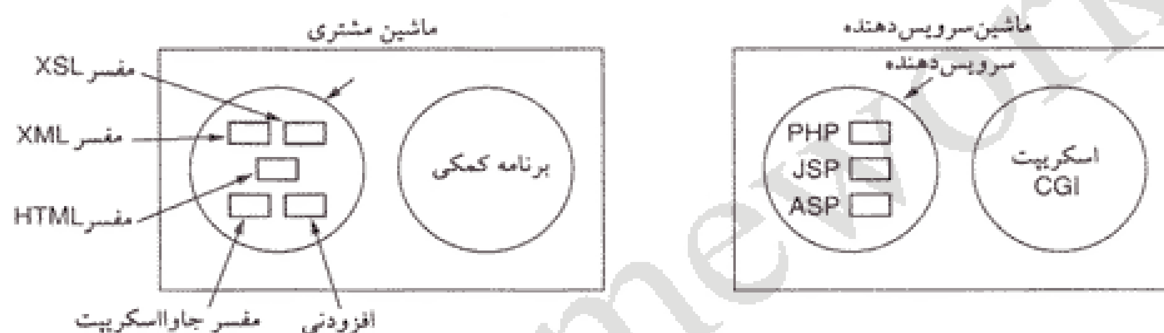
جاوااسکریپت تنها راه نوشتن صفحات وب تعاملی نیست؛ روش دیگر استفاده از اپلت (applet) است. اپلت عبارتست از یک برنامهٔ کوچک جاوا، که برای یک ماشین مجازی بنام JVM (ماشین مجازی جاوا - Java Virtual Machine) کامپایل می‌شود. این اپلت‌ها را می‌توان (با استفاده از برچسب <applet>) در صفحات وب قرار داد، و مرورگرهایی که JVM را داشته باشند، آنها را اجرا خواهند کرد. از آنجائیکه اپلت‌های جاوا بطور مستقیم اجرا نمی‌شوند، مفسر جاوا می‌تواند جلوی اعمال مخرّب آنها را بگیرد؛ حداقل در تئوری که اینطور گفته می‌شود. اما در عمل برنامه‌نویسان باهوش جاوا نارسایی‌های بیشماری در سیستم I/O جاوا شناسایی کرده‌اند، که می‌توان به کمک آنها در سیستم قربانی نفوذ کرد.

میکروسافت در پاسخ به اپلت‌های جاوا (که از ابداعات شرکت سان میکروسیستمز است) کنترل اکتیوایکس (ActiveX control) را معرفی کرد. کنترلهای اکتیوایکس برنامه‌های کامپایل شدهٔ پنتیوم هستند، که مستقیماً روی سخت‌افزار کامپیوتر مشتری اجرا می‌شوند. سرعت و توانائیهای کنترلهای اکتیوایکس بسیار بیشتر و بهتر از اپلت‌های جاواست، چون آنها برنامه‌های واقعی هستند. وقتی مرورگر اینترنت اکسپلورر در صفحهٔ وب به یک کنترل اکتیوایکس برخورد می‌کند، آنرا بار کرده، و (پس از تعیین هویت) اجرا می‌کند. اما، همانطور که می‌توان حدس زد، بار کردن و اجرای برنامه‌های خارجی مشکلات امنیتی زیادی بهمراه دارد، که در فصل آینده به آنها اشاره خواهیم کرد.

از آنجائیکه برنامه‌های جاوا و جاوااسکریپت روی (تقریباً) تمام مرورگرها اجرا می‌شوند، بهترین گزینه برای نوشتن صفحات تعاملی وب هستند، ولی اگر فقط مرورگرهای میکروسافت را هدف قرار داده‌اید، اکتیوایکس هم

به لیست امکانات شما اضافه خواهد شد. بطور کلی، نوشتن برنامه‌های جاوااسکریپت از همه ساده‌تر است، اپلت‌های جاوا سریعتر اجرا می‌شوند، و کنترلهای اکتیوایکس از هر دوی آنها سریعترند. از طرف دیگر، از آنجائیکه اکثر قریب به اتفاق مرورگرها از یک JVM یکسان استفاده می‌کنند (در حالیکه پیاده‌سازی جاوااسکریپت در هیچ دو مرورگری یکسان نیست)، قابلیت انتقال اپلت‌های جاوا بیشتر از برنامه‌های جاوااسکریپت است. در زمینهٔ برنامه‌نویسی جاوااسکریپت کتابهای متعدد و مفصلی وجود دارد، که از میان آنها می‌توان به (Easttom, 2001; Harris, 2001; and McFedries, 2001) اشاره کرد.

قبل از خاتمه دادن به بحث صفحات وب دینامیک، اجازه دهید یک بار دیگر آنچه را تا اینجا دیدیم مرور کنیم. با استفاده از اسکریپت‌ها می‌توان صفحات وب را بصورت کاملاً دینامیک در سرویس‌دهنده ایجاد کرد. این صفحات در کامپیوتر مشتری درست مثل صفحات معمولی HTML نمایش داده می‌شوند، و هیچ تفاوتی با آنها نخواهند داشت. اسکریپت‌ها را می‌توان با پرل، PHP ، JSP ، یا ASP نوشت (شکل ۷-۴۰ را ببینید).



شکل ۷-۴۰. روشهای مختلف ایجاد محتویات دینامیک و نمایش آنها.

تولید محتویات دینامیک در سمت مشتری نیز امکانپذیر است. صفحات وب را می‌توان با XML نوشت، و سپس برای تبدیل آنها به HTML از فایلهای XSL استفاده کرد. با برنامه‌های جاوااسکریپت هم می‌توان محاسبات موردنیاز را انجام داد. و بالاخره، با استفاده از برنامه‌های کمکی می‌توان محتویات صفحه را بطرق مختلف فرمت کرد.

۷-۳-٤ پروتکل انتقال آبرمتن - HTTP

پروتکل انتقال در سراسر وب HTTP (پروتکل انتقال آبرمتن - HyperText Transfer Protocol) است. این پروتکل مشخص می‌کند که مشتری چه چیزهایی می‌تواند به سرویس‌دهنده بفرستد، و سرویس‌دهنده چه پاسخهایی می‌تواند به آن بدهد. درخواست‌ها از نوع متنی (ASCII) هستند، و پاسخ سرویس‌دهنده یکی از انواع RFC 822 MIME . تمام مشتری‌ها و سرویس‌دهنده‌ها باید از این پروتکل پیروی کنند. پروتکل HTTP در RFC 2616 تعریف شده است. در این قسمت مهمترین ویژگیهای این پروتکل را مورد بررسی قرار خواهیم داد.

اتصال

مرورگرها معمولاً از طریق اتصال TCP به پورت 80 سرویس‌دهنده با آن ارتباط برقرار می‌کند، اگرچه این الزامی رسمی نیست. خوبی اتصال TCP آنست که مرورگر یا سرویس‌دهنده هیچکدام لازم نیست نگران گم شدن پیامها، پیامهای تکراری یا خیلی بلند، و یا برگرداندن تصدیق دریافت باشند، چون تمام این کارها را TCP انجام می‌دهد.

در HTTP 1.0 ، بعد از برقراری اتصال یک درخواست فرستاده شده و یک پاسخ دریافت می‌شود، و پس از آن اتصال قطع خواهد شد. در آن زمانی که صفحات HTML فقط متن بودند، این روش کاملاً کفایت می‌کرد. اما

خیلی زود صفحات وب پر شد از تصویر، آیکون و چیزهایی مانند آن، که برقرای یک اتصال TCP برای انتقال هر کدام از آنها اصلاً مقرون بصرفه نبود.

برای حل این مشکل HTTP 1.1 عرضه شد، که از **اتصال پایدار** (persistent connection) پشتیبانی می‌کرد. اتصال پایدار، اتصالیست که می‌توان روی آن چندین بار درخواست و پاسخ ردوبدل کرد. بدین ترتیب، سربارهٔ TCP برای هر صفحهٔ وب بسیار کمتر خواهد شد. در این روش امکان استفاده از تکنیک **خط‌لوله** (pipeline – ارسال درخواست ۲ قبل از برگشت پاسخ درخواست ۱) هم وجود دارد، که سرعت بار کردن صفحات را بسیار بهتر می‌کند.

متد

با اینکه HTTP برای استفاده در وب طراحی شد، اما طراحان آن (هوشمندانه) نگاهی به آیندهٔ برنامه‌های شیء-گرا نیز داشتند. بهمین دلیل در HTTP عملیاتی غیر از درخواست صفحات وب نیز پیش‌بینی شده است، که به آنها **متد** (method) می‌گویند. از همین جاست که پروتکل SOAP امکان وجود پیدا کرد. هر درخواست HTTP یک (یا چند) خط متن ASCII است، که با نام متد شروع می‌شود. در شکل ۷-۴۱ متدهایی که HTTP پشتیبانی می‌کند، را می‌بینید. اضافه کردن متدهای جدید به HTTP (برای انجام کارهای جدید) نیز امکانپذیر است. نام متدها در HTTP به نوع حروف حساس است، بنابراین متد *GET* را نمی‌توان بصورت *get* نوشت.

متد	توضیح
GET	تقاضای خواندن صفحه وب
HEAD	تقاضای خواندن سر صفحه وب
PUT	تقاضای ذخیره کردن صفحه وب
POST	اضافه کردن به یک منبع نام‌دار (یعنی صفحه وب)
DELETE	حذف صفحه وب
TRACE	برگرداندن درخواست ورودی
CONNECT	برای آینده رزرو شده است
OPTIONS	جستجوی گزینه‌های خاص

شکل ۷-۴۱. متدهای HTTP .

برای درخواست ارسال یک صفحهٔ وب از سرویس‌دهنده (یا هر شیء دیگر، و بعبارت کلی‌تر، یک فایل) از متد *GET* استفاده می‌کنیم. صفحهٔ ارسال شده بصورت MIME کُد می‌شود. بیشترین درخواستها از سرویس‌دهنده‌های وب همین متد *GET* است، که شکل کلی آن چنین است:

```
GET filename HTTP/1.1
```

که در آن filename نام شیء موردنظر، و 1.1 ویرایش پروتکل مورداستفاده است.

متد *HEAD* فقط سرآیند صفحه (و نه خود آن) را درخواست می‌کند. از این متد می‌توان برای تست به روز بودن صفحات، و یا تست معتبر بودن URL ها استفاده کرد.

متد *PUT* عکس *GET* است: این متد یک صفحه را به سرویس‌دهنده می‌فرستد. محتویات صفحهٔ موردنظر (که می‌تواند بصورت MIME کُد شده باشد) در بدنهٔ متد *PUT* می‌آید؛ در این متد سرآیند احرازهویت (authentication) – برای اثبات اینکه مشتری اجازهٔ نوشتن محتویات در سرویس‌دهنده را دارد – نیز می‌تواند وجود داشته باشد.

متد *POST* تا حدی شبیه *PUT* است، با این تفاوت که محتویات جدید به انتهای داده‌های قبلی اضافه می‌شود. از این متد اغلب برای ارسال پیام به گروههای خبری یا سیستمهای BBS (Bulletin Board System) استفاده می‌شود. در عمل، متدهای *PUT* و *POST* کاربرد بسیار محدودی دارند.

متد *DELETE* همان کاری را می‌کند، که از آن انتظار می‌رود: حذف صفحه (البته برای این کار هم مانند *PUT* داشتن مجوز ضروریست). هیچ تضمینی نیست که متد *DELETE* با موفقیت انجام شود، چون بسیار امکان دارد که وقتی سرویس‌دهندهٔ HTTP می‌خواهد صفحه را حذف کند، فایل در وضعیتی باشد که حذف آن ممکن نباشد.

متد *TRACE* برای دیباگ بکار می‌رود: این متد به سرویس‌دهنده می‌گوید که درخواست را به همان شکلی که دریافت کرده، برگرداند. در مواردی که یک درخواست بدرستی پاسخ داده نمی‌شود، و می‌خواهیم بدانیم علت آن چیست، متد *TRACE* می‌تواند نشان دهد که سرویس‌دهنده درخواست را چگونه دریافت کرده است.

متد *CONNECT* در حال حاضر استفاده نمی‌شود، و برای آینده پیش‌بینی شده است.

متد *OPTIONS* اجازه می‌دهد تا مشتری اطلاعاتی از سرویس‌دهنده (یا یکی از فایلهای آن) بدست آورد.

هر درخواست یک پاسخ شامل یک خط وضعیت و احتمالاً مقداری اطلاعات دیگر (که می‌تواند صفحهٔ وب یا بخشی از آن باشد) دریافت می‌کند. **خط وضعیت** (status line) شامل یک کُد سه رقمی است که نشان می‌دهد آیا درخواست انجام شده، و اگر نشده، چرا. رقم اول این کُد برای دسته‌بندی پاسخها بکار می‌رود؛ همانطور که در شکل ۷-۴۲ می‌بینید، پنج نوع پاسخ وجود دارد. کُدهای 1xx در عمل بندرت مورد استفاده قرار می‌گیرند. کُدهای 2xx می‌گویند که درخواست پذیرفته شده، و اطلاعات خواسته شده در حال فرستاده شدن است. کُدهای 3xx مشتری را بجای دیگری (یک URL دیگر، یا حافظهٔ نهان خود مشتری) حواله می‌دهند. کُدهای 4xx حاکی از عدم موفقیت درخواست (به علت خطا در درخواست مشتری، یا عدم وجود صفحهٔ خواسته شده) هستند. کُدهای 5xx هم حاکی از آن هستند که خود سرویس‌دهنده با خطا مواجه شده است (خطا در اجرای کُد خواسته شده، یا ترافیک بیش از حد و عدم توانایی در پاسخ به موقع).

سرآیند پیام

معمولاً بدنبال خط درخواست (خطی که مثلاً متد *GET* در آن آمده) خط‌های دیگری (با اطلاعات بیشتر) نیز می‌آیند. به این خط‌ها (که می‌توان آنها را شبیه پارامتر دانست) **سرآیند درخواست** (request header) می‌گویند. پاسخها هم می‌توانند **سرآیند پاسخ** (response header) داشته باشند. برخی از این سرآیندها (که تعدادی از آنها را در شکل ۷-۴۳ می‌بینید) در هر دو جهت کاربرد دارند.

کد	مفهوم	مثال
1xx	Information	۱۰۰ = سرویس دهنده با درخواست مشتری موافق است
2xx	Success	۲۰۰ = درخواست موفق بوده است ، ۲۰۴ = محتویات موجود نیست
3xx	Redirection	۳۰۱ = صفحه جابجا شده ، ۳۰۴ = صفحه موجود در حافظه نهان همچنان معتبر است
4xx	Client error	۴۰۳ = صفحه ممنوع ، ۴۰۴ = پیدا نشد
5xx	Server error	۵۰۰ = خطای داخلی سرویس دهنده ، ۵۰۳ = دوباره سعی کنید

شکل ۷-۴۲. کُدهای پاسخ سرویس‌دهنده.

با سرآیند *User-Agent* مشتری می‌تواند اطلاعاتی دربارهٔ مرورگر و سیستم عامل خود (و چیزهایی از این قبیل) به سرویس‌دهنده بدهد. در اسکریپت شکل ۷-۳۴ دیدید که سرویس‌دهنده چگونه می‌تواند با استفاده از این سرآیند اطلاعات جالبی از مشتری نمایش دهد.

چهار سرآیند *Accept* به سرویس‌دهنده می‌گویند که مشتری چه چیزهایی را می‌تواند بپذیرد. اولین *Accept* نوع MIME موردپذیرش مشتری (مثلاً، *text/html*) را مشخص می‌کند، و دومی مجموعه کاراکتر آن را (مثلاً، ISO-8859-5 یا Unicode-1-1). سومین *Accept* نوع فشرده‌سازی (مثل، *gzip*)، و چهارمی زبان مشتری (مثلاً، ایتالیایی یا اسپانیایی) را اعلام می‌کنند (تا اگر سرویس‌دهنده امکان انتخاب زبان را فراهم کرده باشد، صفحهٔ مناسب را بفرستد). اگر سرویس‌دهنده نتواند خواسته‌های مشتری را اجابت کند، یک کُد خطا برمی‌گرداند.

سرآیند	نوع	محتویات
User-Agent	درخواست	اطلاعات درباره مرورگر و سیستم عامل آن
Accept	درخواست	نوع صفحاتی که مشتری می تواند بپذیرد
Accept-Charset	درخواست	مجموعه کارکترهای قابل قبول مشتری
Accept-Encoding	درخواست	کد گذاری های صفحه قابل قبول مشتری
Accept-Language	درخواست	زبانهای طبیعی قابل قبول مشتری
Host	درخواست	نام DNS سرویس‌دهنده
Authorization	درخواست	کوکی را به سرویس‌دهنده باز پس می فرستد
Cookie	درخواست	لیست احراز هویت مشتری
Date	هردو	زمان و تاریخ ارسال پیام
Upgrade	هردو	پروتکلی که فرستنده می خواهد به آن ارتقاء دهد
Server	پاسخ	اطلاعاتی درباره سرویس دهنده
Content-Encoding	پاسخ	نحوه کدگذاری محتویات صفحه
Content-Language	پاسخ	زبان طبیعی صفحه
Content-Length	پاسخ	طول صفحه (بر حسب بایت)
Content-Type	پاسخ	نوع MIME صفحه
Last-Modified	پاسخ	زمان و تاریخ آخرین تغییر صفحه
Location	پاسخ	فرمان به مشتری برای ارسال درخواست به جای دیگر
Accept-Ranges	پاسخ	سرویس دهنده محدوده بایت را پذیرفت
Set-Cookie	پاسخ	سرویس دهنده از مشتری می خواهد یک کوکی ذخیره کند

شکل ۷-۴۳. چند سرآیند پیام HTTP.

سرآیند *Host* نام سرویس‌دهنده را مشخص می‌کند، و از URL گرفته می‌شود. این سرآیند اجباریست، چون ممکنست چندین نام DNS دارای یک آدرس IP مشترک باشند، و سرویس‌دهنده باید بداند که درخواست مربوط به کدام میزبان است.

سرآیند *Authorization* برای صفحاتی که حفاظت شده هستند، لازم است (در این قبیل موارد، مشتری باید ثابت کند که اجازهٔ دریافت صفحهٔ خواسته شده را دارد).

با اینکه کوکی‌ها در RFC 2109 تعریف شده‌اند، در RFC 2616 نیز دو سرآیند برای کوکی‌ها در نظر گرفته شده است. مشتری برای ارسال کوکی به سرویس‌دهنده‌ای که قبلاً کوکی را از آن گرفته، از سرآیند *Cookie* استفاده می‌کند. سرآیند *Date* یکی از سرآیندهای دو طرفه است، و برای مشخص کردن زمان ارسال پیام بکار می‌رود. سرآیند *Upgrade* برای تسهیل فرآیند ارتقاء (ویرایشهای ناسازگار) پروتکل HTTP در نظر گرفته شده است. این سرآیند به مشتری (یا سرویس‌دهنده) اجازه می‌دهد تا اعلام کند از چه ویرایشی پشتیبانی می‌کند.

سرآیندهای بعدی همگی خاص سرویس‌دهنده هستند، که در پاسخها از آنها استفاده می‌کند. با اولین آنها، یعنی *Server*، سرویس‌دهنده خود را معرفی کرده و برخی از مشخصاتش را اعلام می‌کند.

چهار سرآیند بعدی، که همگی با *-Content* شروع می‌شوند، به سرویس‌دهنده اجازه می‌دهند تا صفحه‌ای را

که در حال فرستادن آن است، توصیف کند.

سرآیند *Last-Modified* مشخص می‌کند که تاریخ آخرین تغییر صفحه چه زمانی بوده است. این سرآیند نقش مهمی در عملکرد حافظهٔ نهان صفحات وب در مشتری دارد.

سرویس‌دهنده با استفاده از سرآیند *Location* به مشتری می‌گوید که باید به URL دیگری مراجعه کند (احتمالاً برای اینکه صفحهٔ خواسته شده به جای دیگری منتقل شده، یا هدایت مشتری به سرویس‌دهنده‌های کم ترافیک‌تر). در مواردی هم که یک شرکت دارای شعبه‌های متعددی در سراسر دنیاست، با استفاده از این سرآیند مشتری را به سرویس‌دهنده‌ای که خاص منطقهٔ وی در نظر گرفته شده، هدایت می‌کند. سرویس‌دهنده می‌تواند برای تشخیص مکان جغرافیایی مشتری از آدرس IP یا زبان درخواستی وی استفاده کند.

گاهی که یک صفحه خیلی بزرگ باشد، مشتریهای کوچک مایلند آنرا بصورت قطعه قطعه دریافت کنند. برخی از سرویس‌دهنده‌ها می‌توانند تعداد بایتهای درخواستی یک مشتری را پذیرفته، و فقط همان مقدار را برای وی بفرستند. سرویس‌دهنده با سرآیند *Accept-Range* آمادگی خود را برای ارسال جزئی صفحات اعلام می‌کند.

دومین سرآیند کوکی، یعنی *Set-Cookie* ، وسیله‌ایست که سرویس‌دهنده به کمک آن کوکی‌ها را به مشتری می‌فرستد. مشتری ملزم است این کوکی‌ها را ذخیره کرده، و همراه با درخواستهای بعدی به سرویس‌دهنده پس بفرستد.

نمونه‌ای از کاربرد HTTP

از آنجائیکه HTTP یک پروتکل متنی است، هیچ نیازی نیست مرورگر باشید تا بتوانید با یک سرویس‌دهندهٔ وب تماس بگیرید: فقط کافیست یک اتصال TCP به پورت 80 سرویس‌دهنده داشته باشید. اکیداً به خواننده توصیه می‌کنیم این قسمت را شخصاً امتحان کند (البته UNIX برای این منظور بهتر است، چون برخی از سیستمهای دیگر وضعیت اتصال را برنمی‌گردانند). برای شروع فرمان زیر را وارد کنید:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org
close
```

این فرمان یک اتصال TCP به پورت 80 سرویس‌دهندهٔ وب IETF (www.ietf.org) برقرار می‌کند. نتیجهٔ کار به فایل *log* هدایت می‌شود، تا بعداً بتوانیم آنرا بهتر بررسی کنیم. پس از آن فرمان *GET* (بهمراه نام فایل و پروتکل) می‌آید، و خط بعدی سرآیند اجباری *Host* است. خط خالی بعدی نیز اجباریست: این خط خالی به سرویس‌دهنده می‌گوید که ارسال سرآیندها از طرف ما تمام شده است. با فرمان *close* هم به برنامهٔ telnet می‌گوئیم که ارتباط را قطع کند.

فایل *log* یک فایل متنی است، که با هر ادیتوری می‌توان آنرا مشاهده کرد. ابتدای این فایل باید چیزی شبیه شکل ۷-۴۴ باشد (البته اگر IETF تازگیها آنرا عوض نکرده باشد).

سه خط اول خروجی برنامهٔ telnet هستند، نه سرویس‌دهندهٔ www.ietf.org . خط چهارم، که با HTTP/1.1 شروع شده، پاسخ IETF است و می‌گوید که این سرویس‌دهنده مایل است با پروتکل HTTP/1.1 با شما صحبت کند. پس از آن تعدادی سرآیند، و سپس محتویات صفحهٔ www.ietf.org/rfc.html می‌آید. قبلاً همهٔ این سرآیندها را دیده‌اید، بجز دو تا از آنها: سرآیند *ETag* (که یک شمارهٔ منحصربفرد برای شناسایی صفحه، مخصوص حافظهٔ نهان، است)، و *X-Pad* (که جزء سرآیندهای استاندارد نیست، و احتمالاً برای مقابله با مشکلات برخی از مرورگرها بکار می‌آید).

۷-۳-۵ بهبود کارایی

احتمالاً بزرگترین نقطه ضعف وب همان محبوبیت آن است. سرویس‌دهنده‌ها، مسیریابها، و خطوط مخابراتی

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>IETF RFC Page</title>

<script language="javascript">
function url() {
 var x = document.form1.number.value
 if (x.length == 1) {x = "000" + x }
 if (x.length == 2) {x = "00" + x }
 if (x.length == 3) {x = "0" + x }
 document.form1.action = "/rfc/rfc" + x + ".txt"
 document.form1.submit
}
</script>

</head>
```

شکل ۷-۴۴. قسمت ابتدایی خروجی *www.ietf.org/rfc.html* .

امروزه با ترافیک شدیدی روبرو هستند، تا آنجا که خیلی‌ها WWW را World Wide Wait (**انتظاری پایان جهانی**) می‌خوانند. برای مقابله با این مشکل، محققان تکنیکهای مختلفی برای بالا بردن کارایی وب ابداع کرده‌اند. در این قسمت با برخی از این تکنیکها آشنا خواهید شد: حافظهٔ نهان، تکثیر سرویس‌دهنده، و شبکه‌های تحویل محتوا.

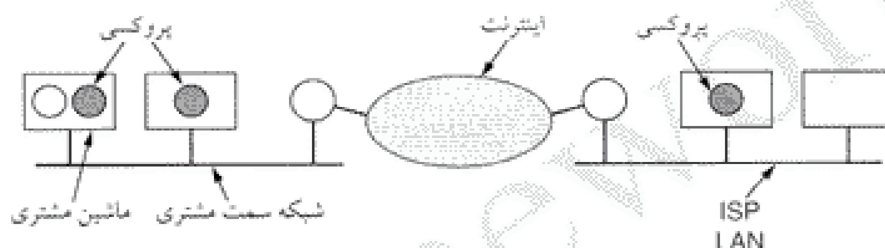
حافظهٔ نهان

یکی از ساده‌ترین روشهای بهبود کارایی وب، ذخیره کردن صفحات برای مراجعهٔ مجدد به آنهاست. این تکنیک بویژه برای صفحاتی که بازدیدکنندهٔ زیادی دارند (مانند، *www.yahoo.com* یا *www.cnn.com*)، مناسب است. ذخیره کردن صفحات وب برای مراجعات بعدی به **حافظهٔ نهان** (caching) معروف است. این حافظهٔ نهان معمولاً از طریق یک واسطه، موسوم به **پروکسی** (proxy) ، بکار گرفته می‌شود، و مرورگر بجای مراجعهٔ مستقیم به سرویس‌دهنده، صفحهٔ موردنظر را از این پروکسی درخواست می‌کند. اگر پروکسی این صفحه را در حافظهٔ نهان خود داشته باشد، بلافاصله آنرا به مرورگر برمی‌گرداند. ولی اگر نداشته باشد، صفحه را از سرویس‌دهنده گرفته، و (بعد از ذخیره کردن در حافظهٔ نهان خود) به مشتری تحویل می‌دهد.

در رابطه با حافظهٔ نهان دو سئوال مهم وجود دارد:

۱. چه کسی باید این کار را انجام دهد؟
۲. صفحات چه مدت باید نگه داشته شوند؟

سئوال اول جوابهای مختلفی می‌تواند داشته باشد. معمولاً PC ها هر کدام یک پروکسی دارند، که به کمک آن می‌توانند صفحاتی را که قبلاً دیده‌اند بسرعت پیدا کنند. در شبکه‌های محلی هم اغلب یک ماشین به این کار اختصاص داده می‌شود، تا اگر یکی از کاربران بخواهد صفحه‌ای را ببیند که کاربر دیگری قبلاً آنرا دیده است، نیازی به مراجعهٔ مستقیم به سرویس‌دهنده نباشد و بتوان آنرا از حافظهٔ نهان پروکسی در اختیار وی گذاشت. اکثر ISP ها هم برای سرعت دادن به دسترسی اینترنت کاربران خود از پروکسی استفاده می‌کنند. همهٔ این پروکسی‌ها همزمان با هم کار می‌کنند، بنابراین درخواستها ابتدا به اولین پروکسی می‌رسد. اگر این پروکسی صفحه را نداشته باشد، سراغ پروکسی شبکهٔ محلی می‌رود، و اگر آن هم صفحه را نداشته باشد، نوبت به پروکسی ISP می‌رسد. این آخری بالاخره هر طور که هست (از حافظهٔ نهان خودش، از حافظهٔ نهان سطح بالاتر، و یا مستقیماً از سرویس‌دهنده) صفحه را به درخواست‌کننده تحویل می‌دهد. به چنین روشی که در آن پروکسی‌ها بصورت زنجیره‌ای به هم متصلند، **حافظهٔ نهان سلسله‌مراتبی** (hierarchical chacing) گفته می‌شود (شکل ۷-۴۵ را ببینید).



شکل ۷-۴۵. حافظهٔ نهان سلسله‌مراتبی با سه پروکسی.

پاسخ سئوال دوم کمی پیچیده‌تر است. برخی از صفحات اصلاً نباید در حافظهٔ نهان ذخیره شوند. برای مثال، صفحه‌ای که قیمت سهام ۵۰ شرکت فعال بورس را نشان می‌دهد، هر ثانیه تغییر می‌کند. اگر چنین صفحه‌ای از حافظهٔ نهان به مشتری داده شود، اطلاعات آن کهنه و غیرقابل‌اعتماد خواهد بود. از طرف دیگر، همین که بازار بورس تعطیل شود، اطلاعات این صفحه تا فردا (که بورس دوباره باز شود) معتبر خواهد بود. همانطور که می‌بینید، قابلیت ذخیره‌سازی یک صفحه در حافظهٔ نهان می‌تواند بشدت به زمان وابسته باشد.

نکتهٔ کلیدی در تعیین مدت نگهداری صفحات در حافظهٔ نهان این است که کاربران تا چه حد می‌توانند کهنگی صفحات را تحمل کنند (از آنجائیکه این صفحه‌ها روی دیسک ذخیره می‌شوند، مقدار آنها چندان اهمیتی ندارد). اگر یک پروکسی صفحات را مدت زیادی در حافظهٔ نهان خود نگه ندارد (و آنها را بسرعت دور بیندازد)، صفحات آن بندرت کهنه می‌شوند، ولی از طرف دیگر کارایی خوبی نیز نخواهد داشت (چون صفحات کمی را از حافظهٔ نهان به مشتری‌ها می‌دهد). اما اگر صفحات را مدت زیادی نگه دارد، اکثر درخواستها را از این حافظهٔ نهان پاسخ می‌دهد، ولی به قیمت کهنه شدن آنها.

دو روش برای حل این مشکل وجود دارد. روش اول از نوعی شهود و گمان برای حدس زدن اینکه هر صفحه چه مدت باید در حافظهٔ نهان نگه داشته شود، استفاده می‌کند. یکی از آنها بر پایهٔ فیلد *Last-Modified* سرآیند صفحه استوار است (شکل ۷-۴۳ را ببینید). اگر صفحه‌ای یک ساعت پیش تغییر کرده باشد، پروکسی آنرا یک ساعت در حافظهٔ نهان خود نگه می‌دارد. اگر صفحه‌ای آخرین بار یک سال پیش تغییر کرده باشد، پیداست که اطلاعات آن بسیار ثابت و پایدار است (مثلاً، صفحه‌ای که دربارهٔ تاریخ روم باستان است)، و می‌توان آنرا یک سال دیگر هم در حافظهٔ نهان نگه داشت، بدون اینکه نگرانی زیادی دربارهٔ کهنه شدن اطلاعات آن وجود داشته باشد. با اینکه روش شهودی فوق غالباً در عمل خوب کار می‌کند، ولی گاهی هم صفحات کهنه برمی‌گرداند.

رهیافت دیگر (که بر اساس یکی از ویژگیهای مدیریت حافظهٔ نهان در RFC 2616 بنا نهاده شده) قدری گرانتر است، ولی احتمال برگرداندن صفحات کهنه را از بین می‌برد. یکی از سودمندترین این ویژگیها سرآیند *If-Modified-Since* است، که پروکسی می‌تواند به سرویس‌دهنده بفرستد. در اینجا پروکسی نام صفحهٔ موردنظر و تاریخ آخرین تغییر آنرا (از سرآیند *Last-Modified*) به سرویس‌دهنده می‌فرستد. اگر صفحهٔ مزبور از این تاریخ به بعد تغییر نکرده باشد، سرویس‌دهنده پیام کوتاهی با مضمون *Not Modified* برمی‌گرداند (کُد وضعیت 304، جدول ۷-۴۲)، که به پروکسی می‌گوید «استفاده از صفحهٔ موجود در حافظهٔ نهان بی‌اشکال است.» اما اگر صفحه بعد از این تاریخ تغییر کرده باشد، سرویس‌دهنده صفحهٔ جدید را برمی‌گرداند. همانطور که می‌بینید، در این روش پروکسی همیشه باید با سرویس‌دهنده تماس بگیرد، که البته در مواقعی که اطلاعات صفحهٔ موجود در حافظهٔ نهان همچنان معتبر باشد، پاسخ سرویس‌دهنده بسیار کوتاه خواهد بود.

این دو روش را می‌توان بآسانی با هم ترکیب کرد. برای مدت زمان ΔT بعد از گرفتن صفحه، پروکسی صفحه را مستقیماً از حافظهٔ نهان به درخواست‌کنندگان می‌دهد. اما پس از آنکه مدتی گذشت، پروکسی با استفاده از پیام *If-Modified-Since* اعتبار صفحه را چک می‌کند. انتخاب ΔT برای هر صفحه هم نیاز به نوعی شهود و گمان دارد، که باز هم به فیلد *Last-Modified* متکیست.

صفحات دینامیک (مانند آنهایی که اسکریپت PHP دارند) هرگز نباید در حافظهٔ نهان ذخیره شوند، چون پارامترهای آنها هر بار تغییر می‌کند. برای حل این مشکل، سرویس‌دهنده با استفاده از یک مکانیزم خاص به تمام پروکسیهایی که بین آن و مشتری قرار دارند، دستور می‌دهد که بدون اطمینان از اعتبار و تازگی محتویات صفحه آنرا به مشتری ندهند. از این مکانیزم برای صفحاتی که محتویات آنها بسرعت تغییر می‌کند، نیز می‌توان استفاده کرد. در RFC 2616 مکانیزمهای مختلفی برای کنترل حافظهٔ نهان تعریف شده است.

روش دیگری که برای بهبود کارایی وب وجود دارد، **حافظهٔ نهان پیشگو** (proactive caching) است. در این روش، وقتی پروکسی صفحه‌ای را می‌آورد، تمام لینکهای موجود در آنرا بررسی کرده، و بطور خودکار آن صفحات را هم بار می‌کند، تا اگر به آنها نیاز شد، از قبل آمادگی داشته باشد. این تکنیک سرعت دسترسی به صفحات را بسیار بالا می‌برد، ولی ترافیک خطوط ارتباطی را هم بشدت افزایش خواهد داد (آن هم برای خواندن صفحاتی که شاید هرگز نیازی به آنها نباشد).

ذخیره کردن صفحات وب در حافظهٔ نهان بهیچوجه موضوع ساده‌ای نیست، و می‌توان ساعتها دربارهٔ آن صحبت کرد. در این زمینه کتابهای متعددی نوشته شده، که از میان آنها می‌توان به (Rabinovich and Spatscheck, 2002; and Wessels, 2001) اشاره کرد. اما اجازه دهید ما این بحث را در همین جا خاتمه دهیم، و سراغ مبحث بعدی برویم.

تکثیر سرویس‌دهنده

حافظهٔ نهان یک تکنیک بهبود کارایی سمت-مشتری است، ولی تکنیکهای سمت-سرویس‌دهنده نیز وجود دارند. یکی از متداولترین این تکنیکها **تکثیر** (replication) محتویات سرویس‌دهنده در نقاط دور از هم است. به این تکنیک گاهی **آینه‌ای‌کردن** (mirroring) نیز گفته می‌شود.

در ساده‌ترین شکل، صفحهٔ اصلی یک سایت آینه‌ای دارای لینکهایی به نقاط مختلف (شمال، جنوب، شرق، و غرب) است. کاربری که روی یکی از این لینکها کلیک می‌کند، با توجه به ناحیه‌ای که در آن زندگی می‌کند، به سرویس‌دهندهٔ مربوطه هدایت می‌شود. از آن پس نیز تمام درخواستهای وی به این سرویس‌دهنده می‌روند.

سایتهای آینه‌ای معمولاً سایتهایی کاملاً ثابت و استاتیک هستند، و برای مدتهای مدید تغییر نمی‌کنند. معمولاً محتویات سایت اصلی کمابیش در سایت‌های آینه‌ای نیز کپی می‌شود (باستثنای آن بخشهایی که منطقاً باید حذف

شوند - برای مثال، مطالب مربوط به بخاری و وسایل گرم‌کننده در مناطق گرمسیر، یا مطالب مربوط به لباس شنا در مناطق قطبی، محلی از اعراب ندارد).

یکی از پدیده‌هایی که متأسفانه در وب بسیار دیده می‌شود، شهرت ناگهانی است: چه بسیار سایتهایی که مدتها ناشناخته، بی‌تماشاچی و راکد بوده‌اند، و ناگهان یک شبه تبدیل به مرکز توجه تمام دنیا شده‌اند. برای مثال، تا روز ششم نوامبر سال ۲۰۰۰ سایت دولت محلی فلوریدا (*www.dos.state.fl.us*) بزحمت روزی چند بازدیدکننده داشت. اما روز هفتم نوامبر که انتخابات ریاست جمهوری ایالات متحده بر سر چند هزار رأی حوزه‌های پرت و دورافتادهٔ این ایالت به جنجال کشیده شد، این سایت تبدیل به یکی از پربیننده‌ترین سایتهای دنیا شد (و حتی برای مدتی بین پنج سایت ردهٔ اول قرار گرفت). لازم به گفتن نیست که این سایت تحمل چنین استقبالی را نداشت، و زیر بار این فشار بسرعت از پا در آمد.

چیزی که یک سایت وب در این قبیل موارد لازم دارد، اینست که بتواند بطور خودکار خود را در محلهای مختلف تکثیر کرده، و تا پایان شرایط اضطراری آنها را فعال نگه دارد، و بعد از عبور طوفان آنها را خاموش کند. البته برای چنین کاری، یک سایت باید از قبل با شرکتهای **میزبانی وب** (Web hosting) هماهنگیهای لازم را انجام داده باشد.

یک استراتژی انعطاف‌پذیرتر ایجاد نسخه‌های تکثیری دینامیک برای تک تک صفحات (بر اساس تقاضاهای رسیده برای هر صفحه) است. در (Pierre et al., 2001; and Pierre et al., 2002) تحقیقاتی که در این زمینه انجام شده، گزارش شده است.

شبکه‌های تحویل محتوا

یکی از نقاط درخشان کاپیتالیسم این است که بالاخره یک نفر فهمید چگونه می‌توان از World Wide Wait (انتظار بی‌پایان برای گرفتن صفحات وب) پول در آورد. چگونه!؟ شرکتهایی بنام CDN (**شبکهٔ تحویل محتوا** - Content Delivery Network) به شرکتهای تولیدکنندهٔ محتوا (مانند سایتهای موسیقی، روزنامه‌ها، و دیگر جاهایی که مایلند محصولات خود را بسرعت به بازار عرضه کنند) پیشنهاد کردند که محصولات آنها را سریعاً و در ازای مبلغی بعنوان حق اشتراک (یا حق مصرف) به دست مصرف‌کنندگان برسانند. بعد از آن که قرارداد بسته شد، صاحب کالا آنرا برای پردازش اولیه و توزیع در اختیار CDN می‌گذارد.

سپس، CDN با تعداد زیادی از ISP ها صحبت کرده، و با آنها (در ازای پرداخت پول) برای در اختیار گرفتن سرویس‌دهنده‌هایی که محتویات در آنها ذخیره خواهد شد، به توافق می‌رسد. این نه تنها منبع درآمدی برای CDN است، بلکه ISP را هم به نان و نوایی می‌رساند، چون مشتریان آن می‌توانند بسرعت به مطالب دلخواهشان دسترسی پیدا کنند (و این در دنیای پر رقابت خدمات اینترنتی، یعنی موفقیت). بهمین دلیل ISP ها برای بستن قرار داد با CDN ها سر و دست می‌شکنند، و بعضی از این CDN ها متجاوز از ۱۰,۰۰۰ سرویس‌دهنده در سراسر دنیا دارند.

وقتی محتویات روی هزاران سرویس‌دهنده پخش شده باشد، کیفیت دسترسی کاربران بشدت بالا خواهد رفت. اما برای آن که این سیستم توزیع‌شده بتواند بخوبی کار کند، باید مکانیزمی برای تغییر مسیر کاربران به نزدیکترین ISP (و ترجیحاً همان ISP که از آن سرویس می‌گیرند) وجود داشته باشد. این تغییر مسیر بایستی بدون هر گونه دستکاری در زیرساخت‌های اینترنت (از قبیل DNS ها) انجام شود. در زیر طرز کار **آکامای** (Akamai) ، بزرگترین CDN دنیا، را بصورت ساده شده بررسی خواهیم کرد.

کار از آنجایی آغاز می‌شود که تولیدکنندهٔ محتوا سایت وب خود را تحویل CDN می‌دهد. در این مرحله، CDN یک پردازش اولیه روی تک تک صفحات سایت انجام داده، و تمام URL های آنرا عوض می‌کند. مدل کاری نهفته در پشت این استراتژی آنست که سایت وب تولیدکنندهٔ محتوا چند صفحهٔ سادهٔ HTML است، که

لینکهایی به فایلهای بزرگتر (مانند تصویر، صدا و ویدئو) دارد. این صفحات تغییر یافته روی سرویس‌دهندهٔ سایت تولیدکنندهٔ محتوا می‌مانند، و فقط فایلهای تصویر، صدا و ویدئو است که به سرویس‌دهنده‌های CDN منتقل می‌شود.

برای درک بهتر روش کار، صفحهٔ اصلی سایت وب Furry Video (شکل ۷-۴۶ الف) را در نظر بگیرید. این صفحه بعد از پردازش اولیه توسط CDN به شکل ۷-۴۶ (ب) در می‌آید، و با نام *www.furryvideo.com/index.html* در سرویس‌دهندهٔ Furry Video ذخیره می‌شود.

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Furry Video's Product List </h1>
<p> Click below for free samples. </p>
<a href="bears.mpg"> Bears Today </a> <br>
<a href="bunnies.mpg"> Funny Bunnies </a> <br>
<a href="mice.mpg"> Nice Mice </a> <br>
</body>
</html>
```

(الف)

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Furry Video's Product List </h1>
<p> Click below for free samples. </p>
<a href="http://cdn-server.com/furryvideo/bears.mpg"> Bears Today </a> <br>
<a href="http://cdn-server.com/furryvideo/bunnies.mpg"> Funny Bunnies </a> <br>
<a href="http://cdn-server.com/furryvideo/mice.mpg"> Nice Mice </a> <br>
</body>
</html>
```

(ب)

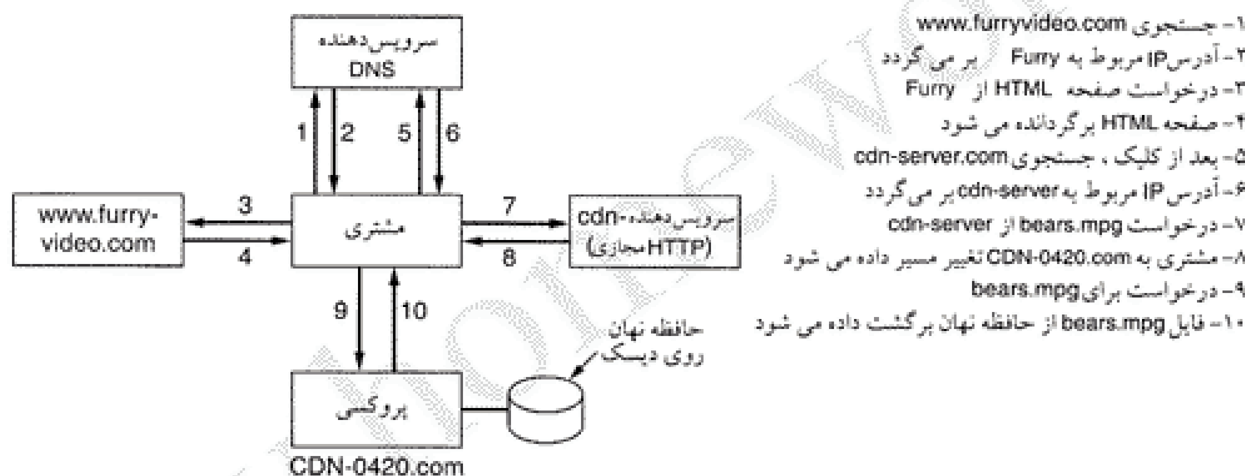
شکل ۷-۴۶. (الف) صفحهٔ وب اولیه. (ب) همان صفحه پس از تغییر در CDN .

وقتی کاربر URL این سایت را وارد می‌کند، DNS طبق روال معمول آدرس IP سایت *www.furryvideo.com* را برمی‌گرداند، و مرورگر صفحه *index.html* را از سرویس‌دهندهٔ Furry Video می‌خواند. اما وقتی روی هر یک از لینکهای این صفحه کلیک شود، مرورگر آدرس سرویس‌دهندهٔ *cdn-server.com* را از DNS می‌پرسد، فایل موردنظر را از این سرویس‌دهنده درخواست می‌کند، و منتظر می‌ماند تا سرویس‌دهندهٔ *cdn-server.com* این فایل را برگرداند.

اما این اتفاق نمی‌افتد، چون *cdn-server.vom* چنین فایلی را ندارد. در اینجا CDN نقش یک سرویس‌دهندهٔ HTTP تقلبی را بازی می‌کند، و با بررسی درخواست رسیده می‌فهمد که تقاضا مربوط به کدام صفحه از کدام تولیدکنندهٔ محتواست. همچنین با بررسی آدرس IP درخواست کننده (و جستجو در پایگاه اطلاعاتی خود) در می‌یابد که کاربر در کدام ناحیهٔ جغرافیایی قرار دارد. با داشتن این اطلاعات، CDN می‌تواند تصمیم بگیرد که کدام سرویس‌دهندهٔ محتوا مناسبترین گزینه برای کاربر موردنظر است. این تصمیم‌گیری چندان هم که بنظر می‌آید ساده نیست، چون ممکنست نزدیکترین محل از نظر جغرافیایی نزدیکترین محل از نظر توپولوژی شبکه نباشد، و یا نزدیکترین سرویس‌دهنده از نظر توپولوژی شبکه در آن لحظه ترافیک بالایی داشته باشد. بعد از انتخاب

سرویس‌دهندهٔ مناسب، CDN یک پیام با کُد 301 که URL نزدیکترین محل در فیلد *Location* آن مشخص شده، به مشتری برمی‌گرداند. برای این مثال فرض می‌کنیم که URL نزدیکترین محل به مشتری *www.CDN-0420.com/furryvideo/bears.mpg* است. مرورگر پس از دریافت این URL به سراغ آن رفته، و فایل *bears.mpg* را طبق روال معمول می‌خواند.

مراحل کار در شکل ۷-۴۷ نشان داده شده است. اولین مرحله تعیین آدرس IP سایت *www.furryvideo.com* ، و پس از آن آوردن صفحهٔ *index.html* و نمایش آن است. این صفحه سه لینک به *cdn-server.com* دارد (شکل ۷-۴۶ ب). وقتی (مثلاً) لینک اول انتخاب شود، DNS آدرس آنرا جستجو کرده (مرحلهٔ ۵) و برمی‌گرداند (مرحلهٔ ۶). با ارسال درخواست فایل *bears.mpg* به *cdn-server.com* (مرحلهٔ ۷)، به مشتری گفته می‌شود که باید به *CDN-0420.com* برود (مرحلهٔ ۸). وقتی مشتری به این محل مراجعه می‌کند (مرحلهٔ ۹)، پروکسی *CDN-0420.com* فایل مزبور را به وی تحویل می‌دهد (مرحلهٔ ۱۰). آنچه که باعث می‌شود این مکانیزم کار کند، مرحلهٔ ۸ است: جایی که یک سرویس‌دهندهٔ HTTP قلابی مشتری را به نزدیکترین پروکسی CDN تغییر مسیر می‌دهد.



شکل ۷-۴۷. مراحل پیدا کردن URL وقتی پای CDN در میان است.

سرویس‌دهندهٔ CDN (که مشتری از آنجا سر در می‌آورد) معمولاً یک پروکسی با حافظهٔ نهان بسیار بزرگ است (که قسمت اعظم محتویات در آن قرار دارند). با این تمهید (استفاده از پروکسی بجای یک سرویس‌دهندهٔ وب معمولی) کارایی سیستم بنحو قابل توجهی بالا می‌رود. برای کسب اطلاعات بیشتر دربارهٔ شبکه‌های تحویل محتوا به (Hull, 2002; and Rabinovich and Spatscheck, 2002) مراجعه کنید.

۷-۳-۶ وب بیسیم

این روزها تقاضای زیادی برای دستگاههای کوچک بیسیم که توانایی کار با وب را داشته باشند، وجود دارد. در حقیقت، اولین قدمهای تجربی در این زمینه قبلاً برداشته شده است. شکی نیست که در سالهای آینده تغییرات زیادی را در این زمینه شاهد خواهیم بود، ولی جا دارد که ایده‌های اساسی وب بیسیم را بررسی کنیم، تا دریابیم کجا هستیم و به کجا می‌رویم. در این قسمت بررسی خود را روی دو سیستمی که زودتر از همه به بازار آمدند، متمرکز خواهیم کرد: WAP و I-Mode .

WAP

با رواج روزافزون اینترنت و تلفن همراه، دور نبود روزی که ایدهٔ ترکیب این دو تکنولوژی مطرح شود. ایدهٔ چنین سیستمی اولین بار توسط کنسرسیومی از شرکتهای نوکیا، اریکسون، موتورولا، و phone.com (Unwired

Planet سابق) پیش کشیده شد، و اکنون صدها شرکت دیگر آنرا پشتیبانی می‌کنند. این سیستم اکنون با نام WAP (**پروتکل کاربردهای بیسیم** – Wireless Application Protocol) شناخته می‌شود.

دستگاه WAP می‌تواند یک تلفن همراه پیشرفته، یک PDA ، و یا یک کامپیوتر سفری ساده باشد (استاندارد WAP محدودیتی برای نوع دستگاه قائل نشده است). WAP در واقع به زیرساختهای بیسیم دیجیتال موجود متکی است. کاربر می‌تواند از طریق لینکهای بیسیم به دروازهٔ WAP (WAP Gateway) وصل شده، و درخواست خود برای صفحات وب را ارسال کند. اولین جایی که برای این درخواست چک می‌شود، حافظهٔ نهان دروازه است: اگر صفحه در حافظهٔ نهان دروازه موجود باشد، به کاربر برگردانده می‌شود؛ اگر نباشد، از اینترنت (که ارتباط دروازه با آن از طریق کابل یا فیبر است) گرفته شده، و به کاربر داده می‌شود. به این ترتیب، WAP 1.0 اساساً یک سیستم سوئیچینگ مداری بود که هزینهٔ آن بر اساس زمان مکالمه دریافت می‌شد؛ البته مصرف‌کنندگان هم از چنین چیزی (آن هم برای دیدن صفحات وب روی مانیتور کوچک تلفنهای همراه) خوششان نیامد، و WAP 1.0 شکست خورد (که البته این شکست علتهای دیگری هم داشت). با این حال، ایدهٔ WAP (و رقیب آن، I-Mode) هنوز شکست نخورده، و بنظر می‌رسد WAP 2.0 بتواند با موفقیت همراه باشد. از آنجائیکه WAP 1.0 اولین تلاش برای پیاده‌سازی اینترنت بیسیم بود، نگاه مختصری به آن نمی‌تواند خالی از فایده باشد.

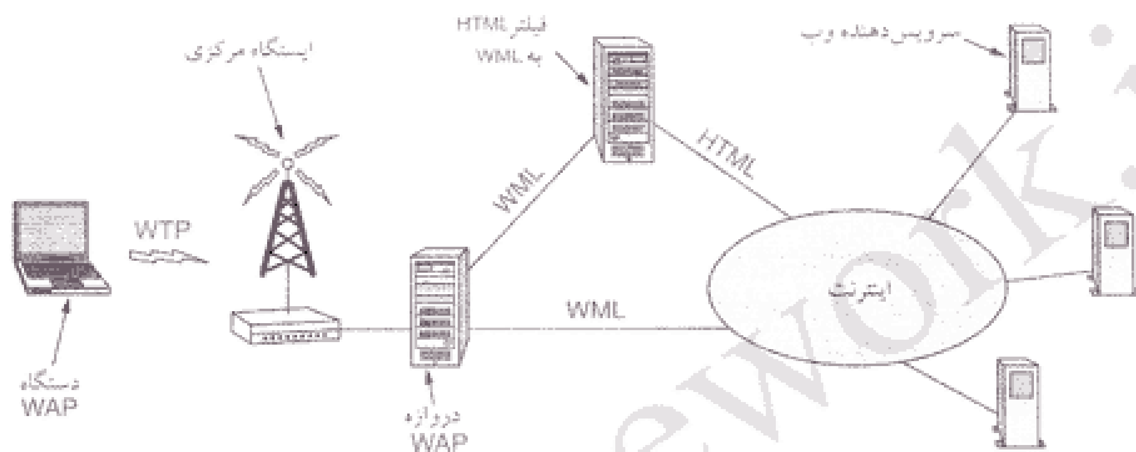
WAP اساساً یک پشتهٔ پروتکل است برای دسترسی وب، که برای وسایلی با پهنای باند کم، CPU کُند، مقدار حافظهٔ کم، و مانیتور کوچک بهینه شده است. این رهیافت که آشکارا با استاندارد PC های امروزی متفاوت است، منجر به پروتکلی متفاوت نیز شده است. لایه‌های پروتکل WAP را در شکل ۷-۴۸ ملاحظه می‌کنید.

محیط برنامه های کاربردی بی‌سیم
پروتکل نشست بی‌سیم
پروتکل تبادل بی‌سیم
ایمنی لایه انتقال بی‌سیم
پروتکل دیتاگرام بی‌سیم
لایه منتقل کننده (GSM, CDMA, D-AMPS, GPRS, etc.)

شکل ۷-۴۸. پشتهٔ پروتکل WAP .

پائینترین لایه (و در واقع لایهٔ فیزیکی) از تمام سیستمهای تلفن همراه موجود (از جمله GSM ، D-AMPS ، و CDMA) پشتیبانی می‌کند. در WAP 1.0 نرخ داده 9600 bps است. بالای این لایه پروتکل دیتاگرام، WDP (**پروتکل دیتاگرام بیسیم** – Wireless Datagram Protocol)، که اساساً همان UDP است، قرار دارد. پس از آن لایه‌ای برای ایمنی داده‌ها (که در مخابرات بیسیم الزامیست) می‌آید. این لایه، WTLS (**ایمنی لایهٔ انتقال بیسیم** – Wireless Transport Layer Security)، زیرمجموعه‌ایست از SSL (که از ابداعات نت‌اسکیپ می‌باشد). سپس لایهٔ تبادل، WTP (**پروتکل تبادل بیسیم** – Wireless Transaction Protocol)، می‌آید که وظیفهٔ آن مدیریت درخواستها و پاسخهاست. این لایه جایگزین TCP شده، که بدلیل کارایی پائین نمی‌توان از آن در ارتباطات بیسیم استفاده کرد. پس از آن یک لایهٔ نشست می‌آید، که شبیه HTTP/1.1 است، ولی بمنظور کارایی بهتر تغییراتی در آن صورت گرفته است. و بالاخره، در بالای همهٔ اینها یک مینی مرورگر – در لایهٔ WAE (**محیط کاربردی بیسیم** – Wireless Application Environment) – قرار می‌گیرد.

بدون شک یکی از علل عدم پذیرش WAP (علاوه بر هزینهٔ بالا) این واقعیت است که WAP از HTML استفاده نمی‌کند: WAE از یک زبان علامتگذاری خاص بنام WML (زبان علامتگذاری بیسیم - Wireless Markup Language) استفاده می‌کند، که به XML متکی است. در نتیجه، یک دستگاه WAP فقط صفحاتی را می‌تواند بخواند که قبلاً به WML تبدیل شده باشند. از آنجائیکه این ویژگی کاربرد WAP را بشدت محدود خواهد کرد، در معماری WAP فیلترهایی برای تبدیل آنی HTML به WML در نظر گرفته شده است (شکل ۷-۴۹ را ببینید).



شکل ۷-۴۹. معماری WAP .

WAP ، با وجود تمام شایستگی‌هایش، شاید کمی از زمانه جلوتر بود. وقتی WAP برای اولین بار راه‌اندازی شد، خارج از W3C کمتر کسی XML را می‌شناخت، و بهمین دلیل همه جا فریاد زدند: «**WAP از HTML پشتیبانی نمی‌کند.**» شاید بهتر بود می‌گفتند: «**WAP از استاندارد جدید HTML پشتیبانی می‌کند.**» ولی وقتی ضربه وارد آمد، دیگر برای جبران آن دیر شده بود، و WAP 1.0 هرگز نتوانست دوباره کمر راست کند. قبل از اینکه داستان WAP را ادامه دهیم، نگاهی به نزدیکترین رقیب آن، یعنی I-Mode ، خواهیم داشت.

I-Mode

در همان زمانیکه کنسرسیوم چندملیتی شرکتهای کامپیوتری و مخابراتی در تلاش بودند تا یک استاندارد باز برای HTML توسعه دهند، ژاپنی‌ها بیکار ننشسته بودند. در آنجا خانمی بنام ماری ماتسوناگا رهیافت جدیدی برای وب بیسیم اختراع کرد، و نام آنرا I-Mode (Information-Mode) گذاشت. او توانست شرکت مخابرات بیسیم ژاپن (که زیرمجموعهٔ وزارت تلفن ژاپن محسوب می‌شد) را متقاعد کند که اختراعش مفید است، و در فوریه ۱۹۹۹ NTT DoCoMo (که به زبان ژاپنی معادل عبارت «**شرکت تلفن و تلگراف ژاپن: هرجا که هستید**» است) رسماً شروع بکار کرد. بعد از سه سال، این سرویس اکنون متجاوز از ۳۵ میلیون مشترک دارد، که به ۴۰,۰۰۰ سایت وب I-Mode دسترسی دارند. مدیران این سیستم (در غیاب WAP) بیشترین لاف‌ها را دربارهٔ موفقیت مالی آن زده‌اند. اما اجازه دهید ببینیم I-Mode چیست و چگونه کار می‌کند.

سیستم I-Mode دارای سه مؤلفهٔ اصلی است: یک سیستم انتقال جدید، یک گوشی جدید، و یک زبان جدید برای طراحی صفحات وب. سیستم انتقال از دو شبکهٔ مجزا تشکیل شده است: شبکهٔ تلفن همراه سوئیچینگ بسته‌ای موجود (که تا حدی شبیه D-AMPS است)، و یک شبکهٔ سوئیچینگ مداری که اختصاصاً برای سرویس I-Mode ایجاد شده است. سرویس I-Mode از شبکهٔ سوئیچینگ بسته‌ای استفاده می‌کند و ارتباط آن دائمی

است (مانند ADSL یا کابل). بنابراین چیزی بنام هزینهٔ اتصال در آن وجود ندارد، و بجای آن هزینهٔ مشترک بر اساس بسته‌های فرستاده شده محاسبه می‌شود. در حال حاضر امکان استفادهٔ همزمان از هر دو شبکه وجود ندارد.

گوشی I-Mode شبیه گوشی‌های معمولی تلفن همراه است، با یک صفحهٔ مانیتور بزرگتر. حتی NTT DoCoMo در تبلیغات خود دستگاههای I-Mode را نسل جدید و پیشرفتهٔ تلفنهای همراه معرفی می‌کند، نه ترمینالهای بیسیم وب (چیزی که در واقع هستند). بسیاری از مشترکان این سیستم حتی نمی‌دانند که روی اینترنت هستند. اغلب آنها تصور می‌کنند که دستگاه I-Mode فقط یک تلفن همراه پیشرفته است. از آنجائیکه I-Mode فقط یک سرویس است، کاربران امکان برنامه‌نویسی با گوشی خود را ندارند (با اینکه گوشی آنها از کامپیوترهای سال ۹۵ قویتر است، و احتمالاً حتی می‌تواند ویندوز ۹۵ یا یونیکس را اجرا کند).

وقتی یک گوشی I-Mode روشن می‌شود، فهرستی از سرویسهای رسمی و تأئید شده به کاربر ارائه می‌کند. تعداد این سرویسها بیش از ۱۰۰۰ تاست، که به ۲۰ دسته تقسیم شده‌اند. هر سرویس که در واقع یک سایت I-Mode کوچک است، توسط یک شرکت مستقل اداره می‌شود. برخی از مهمترین دسته‌هایی که در این منو ظاهر می‌شوند، عبارتند از: ایمیل، اخبار، وضع آب و هوا، ورزش، بازی، خرید، نقشه، طالع‌بینی، سرگرمی، مسافرت، راهنمای اعمال مذهبی، انواع زنگهای تلفن، دستورات آشپزی، قمار، بانکداری خانگی، و قیمتهای بورس. این سرویسها تا حد زیادی نوجوانان و جوانان (با سنی حدود ۲۰ سال) را هدف گرفته‌اند، افرادی که عاشق اسباب‌بازیهای الکترونیکی (مخصوصاً اسباب‌بازیهای رنگارنگ) هستند. این موضوع که بیش از ۴۰ شرکت فقط زنگ تلفن می‌فروشند، می‌تواند گویای حقایق زیادی باشد. در اینجا هم محبوبترین سرویس ایمیل است، که اجازه می‌دهد پیامهایی تا ۵۰۰ بایت رد وبدل شود (نسبت به سرویس SMS با محدودیت ۱۶۰ بایتی، پیشرفت چشمگیری محسوب می‌شود). بازی نیز یکی دیگر از سرویسهای پرطرفدار I-Mode است.

بیش از ۴۰,۰۰۰ سایت وب I-Mode نیز وجود دارد، که کاربر باید URL آنها را وارد کند (و از طریق منو قابل دسترسی نیستند). منوی رسمی I-Mode بسیار شبیه دروازه‌های وب (مانند سایت !Yahoo) است، که به کاربر اجازه می‌دهند تا بدون وارد کردن URL و فقط با یک کلیک به سایتهای مختلف دسترسی پیدا کند.

سرویسهای رسمی I-Mode تحت کنترل شدید NTT DoCoMo قرار دارند. برای آن که یک سرویس در منوی رسمی I-Mode قرار گیرد، باید شرایط مختلفی را برآورده کند. برای مثال، یک سرویس نباید تأثیرات منفی اجتماعی داشته باشد، فرهنگهای لغت ژاپنی-انگلیسی باید به مقدار کافی لغت داشته باشند، سرویسهای زنگ تلفن باید بطور مرتب زنگهای جدیدی عرضه کنند، و هیچ سایتی نباید مطالب بیهوده و سبک (یا چیزی علیه NTT DoCoMo) ارائه کند (Frengle, 2002) . از طرف دیگر، سایتهای اینترنتی I-Mode مجازند هر کاری می‌خواهند بکنند.

مدل تجاری I-Mode تفاوت اساسی با اینترنت دارد. هزینهٔ اشتراک I-Mode فقط چند دلار در ماه است. از آنجائیکه در این سیستم هزینه‌ها بر اساس بسته‌های ارسال شده محاسبه می‌شود، در واقع با این شارژ اولیه کار چندانی نمی‌تواند کرد. کاربر می‌تواند هزینهٔ ماهیانهٔ ثابت بیشتری (برای تعداد بستهٔ بیشتر) بپردازد، و در شارژ بسته‌ها صرفه‌جویی کند، چون با بالا رفتن پرداخت ثابت ماهیانه تعداد بسته‌هایی که می‌تواند بفرستد، بصورت تصاعدی بالا خواهد رفت. اگر وسط ماه بسته‌های مجانی شما تمام شود، می‌تواند بصورت بر-خط بسته‌های بیشتری بخرید.

برای استفاده از یک سرویس باید مشترک آن شوید، کاری که فقط با یک کلیک روی آن و وارد کردن کُد PIN خود می‌توانید انجام دهید. اغلب سرویسهای رسمی هزینه‌ای معادل ۱ تا ۲ دلار در ماه دارند. NTT DoCoMo این پول را در صورتحساب تلفن شما منظور می‌کند، و بعد از کسر ۹٪ سهم خود بقیه (۹۱٪) را به شرکت ارائه کنندهٔ سرویس مسترد می‌کند. اگر یک سرویس غیررسمی ۱,۰۰۰,۰۰۰ مشترک داشته باشد، مجبور است هر ماه ۱,۰۰۰,۰۰۰ صورتحساب ۱ تا ۲ دلاری صادر کرده و برای مشتریان خود بفرستد. در حالیکه اگر سرویس خود را

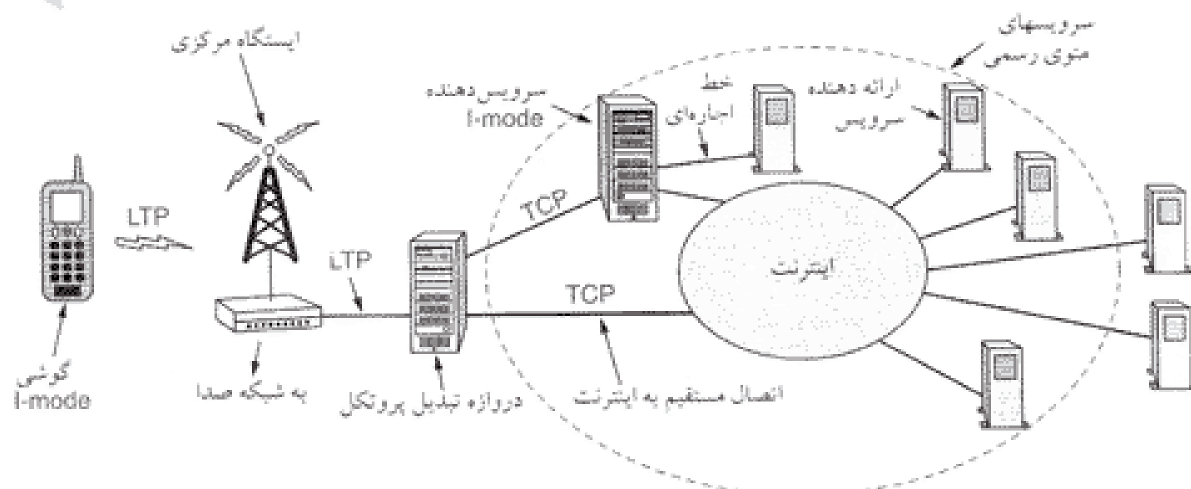
بصورت رسمی در آورد، NTT DoCoMo عملیات بانکی را بجای آن انجام داده و بازای آن ۹۰,۰۰۰ دلار از کل مبلغ کم می‌کند. صرفه‌جویی در هزینه‌های بانکی وصول حق اشتراک از مشتریان انگیزه‌ای بسیار جدی برای تمایل شرکتها به رسمی شدن است (و NTT DoCoMo هم که به حق خود می‌رسد!). رسمی شدن احتمال جذب مشتری را هم افزایش می‌دهد، چون سایت شما وارد منوی اولیهٔ I-Mode خواهد شد. این وسط فقط کاربران هستند که بجای یک صورتحساب باید برای چهار چیز پول بدهند: پول تماسهای تلفنی معمولی، حق اشتراک I-Mode ، حق اشتراک سرویسها، و هزینهٔ بسته‌های اضافی.

علیرغم اقبال گسترده در ژاپن، هنوز مشخص نیست که I-Mode بتواند در اروپا و آمریکا نیز به چنان موفقیتی دست یابد، چون وضعیت اجتماعی ژاپن تفاوت زیادی با جوامع غربی دارد. اول از همه اینکه، اکثر مشتریان بالقوهٔ این سیستم در غرب (نوجوانان، دانشجویان، و کارمندان) دارای PC های بزرگ در خانه با دسترسی اینترنت پُرسرعت (حداقل 56 kbps) هستند، چیزی که در ژاپن چندان مرسوم نیست (اول بعلت فضای کوچک خانه‌های ژاپنی، و دوم بدلیل نرخهای کمرشکن NTT برای سرویسهای تلفن – ۷۰۰ دلار برای نصب یک خط تلفن، و ساعتی ۱/۵ دلار هزینهٔ تماسهای شهری). در ژاپن خیلی از افراد فقط از طریق I-Mode به اینترنت دسترسی دارند.

دوم اینکه، در غرب کسی عادت ندارد برای هر سایتی که می‌خواهد برود، ۱ دلار حق اشتراک بدهد (پرداخت پول برای هر MB خواندن صفحات که دیگر بماند!). اغلب ISP های غربی (تحت فشار مشتریان خود) هزینهٔ دسترسی اینترنت را به یک شارژ ثابت ماهیانه (مستقل از مصرف واقعی آنان) تقلیل داده‌اند.

سوم اینکه، اغلب ژاپنی‌ها در زمان رفت و آمد به محل کار یا مدرسه (و در قطار) از I-Mode استفاده می‌کنند، در حالیکه در اروپا افراد کمی برای رفت و آمد از قطار استفاده می‌کنند (و این پدیده در آمریکا حتی از اروپا هم نادرتر است). استفاده از I-Mode در خانه، آن هم کنار یک مانیتور ۱۷ اینچی، با یک خط 1-Mbps ADSL (بدون اینکه نیازی باشد نگران حجم اطلاعات رد و بدل شده باشید) چندان با عقل سلیم جور در نمی‌آید. با این حال، هیچکس چنین محبوبیتی را برای تلفنهای همراه هم پیش‌بینی نمی‌کرد، پس شاید در غرب هم جایی برای I-Mode باشد.

همانطور که قبلاً هم گفتیم، گوشی‌های I-Mode برای ارتباطات صدا از شبکهٔ موجود سوئیچینگ مداری، و برای ارتباطات داده از یک شبکهٔ سوئیچینگ بسته‌ای جدید استفاده می‌کنند. شبکهٔ داده بر اساس CDMA ، با بسته‌های ۱۲۸ بایتی و با نرخ 9600 bps کار می‌کند (شکل ۷–۵۰ را ببینید). گوشی با استفاده از LPT (**پروتکل سبک انتقال** – Lightweight Transport Protocol) با **دروازهٔ تبدیل پروتکل** (protocol conversion gateway)



شکل ۷–۵۰. ساختار شبکهٔ دادهٔ I-Mode : پروتکلهای انتقال.

ارتباط می‌گیرد. این دروازه توسط یک فیبر نوری پهن-باند به سرویس‌دهندهٔ I-Mode (که به تمام سرویسها متصل است) وصل می‌شود. وقتی کاربر یکی از سرویسهای رسمی را انتخاب می‌کند، این درخواست به سرویس‌دهندهٔ I-Mode فرستاده می‌شود، که (برای بهبود کارایی شبکه) اغلب صفحه‌ها را در حافظهٔ نهان خود دارد. درخواست برای سایتهایی که جزء متوی رسمی نیستند، مستقیماً به اینترنت فرستاده می‌شود.

گوشی‌های فعلی I-Mode با 100-MHz CPU کار می‌کنند، و چندین مگابایت حافظهٔ ROM ، چیزی حدود یک مگابایت RAM و یک صفحهٔ مانیتور کوچک دارند. وضوح مانیتور I-Mode بایستی حداقل 94 × 72 پیکسل باشد، ولی دستگاههایی با وضوح 160 × 120 پیکسل نیز عرضه شده‌اند. این مانیتورها از رنگ 8-bit پشتیبانی می‌کنند، که قادرست 256 رنگ مختلف را نمایش دهد. با اینکه این تعداد رنگ برای نمایش عکس کافی نیست، اما طرح و تصاویر کارتونی را بخوبی نشان می‌دهد. گوشی‌های I-Mode (طبعاً) ماوس ندارند، و حرکت بین آیتمهای صفحه به کمک کلیدها صورت می‌گیرد.

ساختار نرم‌افزاری I-Mode را در شکل ۷-۵۱ ملاحظه می‌کنید. در پائین‌ترین لایه یک سیستم عامل بی‌درنگ (real-time) قرار دارد، که سخت‌افزار را کنترل می‌کند. پس از آن ماژول ارتباط با شبکه می‌آید، که از پروتکل LPT اختصاصی NTT DoCoMo استفاده می‌کند. بالای این لایه یک سیستم مدیریت پنجره (window manager) قرار دارد، که متن و گرافیک‌های ساده (فایلهای GIF) را نمایش می‌دهد. البته با مانیتوری به ابعاد حداکثر 160 × 120 پیکسل چیز زیادی برای مدیریت کردن وجود ندارد.

<table>
<tr><td colspan="3">User interaction module (ماژول تعامل با کاربر)</td></tr>
<tr><td>Plug-ins</td><td>cHTML interpreter</td><td>Java</td></tr>
<tr><td colspan="3">Simple window manager (مدیر پنجره ساده)</td></tr>
<tr><td colspan="3">Network communication (ارتباطات شبکه)</td></tr>
<tr><td colspan="3">Real-time operating system (سیستم عامل زمان واقعی)</td></tr>
</table>

شکل ۷-۵۱. ساختار نرم‌افزاری I-Mode .

لایهٔ چهارم شامل مفسر صفحات وب (یعنی همان مرورگر) است؛ I-Mode فقط از زیرمجموعه‌ای از HTML بنام cHTML (HTML فشرده – compact HTML : که بر اساس HTML 1.0 قرار دارد) پشتیبانی می‌کند. برنامه‌های کمکی و افزودنی (مانند مرورگرهای معمولی) نیز در این لایه قرار می‌گیرند. یکی از برنامه‌های کمکی I-Mode مفسری بر اساس ویرایش اصلاح‌شدهٔ JVM است. و بالاخره، در بالای همهٔ لایه‌ها ماژول تعامل باکاربر (user interaction) قرار گرفته است، که وظیفهٔ آن ارتباط با کاربر از طریق صفحه کلید و مانیتور می‌باشد.

اجازه دهید نگاه دقیقتری به cHTML بیندازیم. همانطور که گفتیم، cHTML تقریباً همان HTML 1.0 است، که برای انطباق با گوشی‌های تلفن همراه تغییراتی در آن صورت گرفته است. استاندارد cHTML از طرف W3C برای نظرخواهی ارائه شده، ولی از آنجائیکه W3C علاقهٔ چندانی به آن نشان نداده، به احتمال زیاد همچنان بصورت محصول اختصاصی NTT DoCoMo باقی خواهد ماند.

اکثر برچسبهای HTML مانند <html> ، <head> ، <title> ، <body> ، <hn> ، <center> ، ، ، <menu> ، ،
 ، <p> ، <hr> ، <img> ، <form> ، و <input> در cHTML هم وجود دارند، ولی <b> و <i> حذف شده‌اند.

یدی بنام *tel* به برچسب <a> (لینک به صفحات دیگر) اضافه شده، که با آن می‌توان با یک شماره تلفن تماس گرفت. از یک جهت *tel* شبیه *mailto* است، که وقتی کاربر آنرا انتخاب می‌کند، مرورگر با اجرای برنامهٔ ایمیل به کاربر اجازه می‌دهد به آدرس مشخص شده پیام بفرستد؛ با این تفاوت که در اینجا مرورگر شماره تلفن را می‌گیرد. بعنوان مثال، با این صفت می‌توان یک دفترچه تلفن تصویری ساخت، که وقتی روی هر یک از تصاویر کلیک می‌کنید، گوشی I-Mode شمارهٔ مربوطه را بگیرد. (URL های تلفنی در RFC 2806 مورد بحث قرار گرفته‌اند.)

مرورگر cHTML محدودیتهای دیگری نیز دارد، مثلاً از جاوااسکریپت، فریم، شیوه‌نامه، و رنگ یا تصویر زمینه پشتیبانی نمی‌کند. مرورگرهای I-Mode تصاویر JPEG را هم نمایش نمی‌دهند، چون قدرت کافی برای خارج کردن سریع این تصاویر از حالت فشرده ندارند. صفحات cHTML می‌توانند اپلت جاوا داشته باشند، ولی اندازهٔ آنها (بدلیل کم بودن سرعت انتقال روی لینکهای هوایی) به 10 KB محدود است.

با اینکه NTT DoCoMo برخی از برچسبهای HTML را حذف کرده، ولی برچسبهای جدیدی را هم به آن اضافه کرده است. یکی از این برچسب‌ها <blink> است، که متن را بصورت چشمک‌زن درمی‌آورد. شاید این برچسب جانشین <b> (که در cHTML حذف شده) باشد، ولی بهر حال این اقدام NTT DoCoMo با استاندارد HTML که به ظاهر صفحات وب بی‌توجه است، همخوانی چندانی ندارد. برچسب جدید دیگر <marquee> است، که متن را در عرض صفحه حرکت می‌دهد.

به برچسب
 نیز صفت جدیدی بنام *align* اضافه شده، که برای نمایش مرتب کلمات روی مانیتورهای ۶سطر × ۱۶ حرف در نظر گرفته شده است. در چنین مانیتورهایی احتمال زیادی هست که کلمات از وسط شکسته شوند (شکل ۷-۵۲الف را ببینید). صفت *align* کمک می‌کند تا این مشکل تا حد زیادی رفع شود (مانند آنچه در شکل ۷-۵۲ب می‌بینید). البته جالبست بدانید که در زبان ژاپنی چیزی بنام شکستن کلمات وجود ندارد، چون هر علامت **کانجی** (kanji - الفبای ژاپنی) - که در یک سلول 10 × 9 پیکسل یا 12 × 12 پیکسل نوشته می‌شود - معادل یک کلمه در زبان انگلیسی است. در زبان ژاپنی یک خط متن می‌تواند از هر نقطه‌ای شکسته شود.

(الف)	(ب)
The time has com e the walrus sai d to talk of man y things. Of sho es and ships and sealing wax of c	The time has come the walrus said to talk of many things. Of shoes and ships and sealing wax

شکل ۷-۵۲. آشفتگی در صفحهٔ مانیتور 6 × 16 .

صفت جد با اینکه زبان ژاپنی دهها هزار کانجی دارد، NTT DoCoMo زبان تصویری جدیدی بنام اموجی (emoji) با ۱۶۶ علامت اختراع کرده است، که تا حدی شبیه خندانک‌های شکل ۷-۶ هستند. در این زبان علامتهای زیبایی برای نمادهای طالع‌بینی، آبجو، همبرگر، پارک تفریحات، روز تولد، تلفن همراه، سگ، گربه، کریسمس، قلب شکسته، بوسه، خلق و خو، خواب، و البته زیبا و دلفریب وجود دارد.

ویژگی دیگر cHTML امکان انتخاب لینکهای صفحهٔ وب با استفاده از صفحه کلید است (که البته در جایی که ماوس وجود ندارد، ویژگی بسیار مهمی است). در شکل ۷-۵۳ نمونه‌ای از یک صفحهٔ cHTML که در آن از این ویژگی استفاده شده، را مشاهده می‌کنید.

با اینکه I-Mode در سمت مشتری با محدودیتهای زیادی مواجه است، در سمت سرویس‌دهنده هیچ

```
<html>
<body>
<h1> Select an option </h1>
<a href="messages.chtml" accesskey="1"> Check voicemail </a> <br>
<a href="mail.chtml" accesskey="2"> Check e-mail </a> <br>
<a href="games.chtml" accesskey="3"> Play a game </a>
</body>
</html>
```

شکل ۷-۵۳. نمونه‌ای از یک فایل cHTML .

محدودیتی ندارد. سرویس‌دهنده‌های I-Mode کامپیوترهای قدرتمندی هستند که از CGI ، Perl ، PHP ، JSP ، ASP (و هر چیزی که یک سرویس‌دهندهٔ وب معمولی می‌تواند داشته باشد) پشتیبانی می‌کنند.

در شکل ۷-۵۴مقایسه‌ای بین سیستمهای نسل اول WAP و I-Mode بعمل آمده است. با آنکه برخی از این تفاوتها کوچک بنظر می‌رسند، ولی در جای خود اهمیت زیادی دارند. برای مثال، اغلب نوجوانان ۱۵ ساله کارت اعتباری ندارند، بنابراین امکان واریز هزینه‌های خرید الکترونیکی به صورتحساب ماهیانهٔ تلفن می‌تواند عامل تعیین‌کننده‌ای در جلب این قبیل مشتریان داشته باشد.

برای کسب اطلاعات بیشتر دربارهٔ I-Mode می‌توانید به (Frengle, 2002; and Vacca, 2002) مراجعه کنید.

ویژگی	WAP	I-mode
ماهیت	پشته پوتکل	سرویس
دستگاه	گوشی، PDA ، کامپیوتری	گوشی
دسترسی	تلفنی	همیشه برقرار
شبکه زیرین	سوئیچینگ مداری	سوئیچینگ مداری و بسته ای
نرخ داده	9600 bps	9600 bps
صفحه نمایش	سیاه و سفید	رنگی
زبان علامتگذاری	WML (XML application)	cHTML
زبان اسکریپت نویسی	WML script	ندارد
هزینه	بر دقیقه	بربسته
پرداخت خریدها	کارت اعتباری	صورتحساب تلفن
علایم تصویری	خیر	بلی
استاندارد سازی	استاندارد باز مجمع WAP	در مالکیت NTT DoCoMo
محل استفاده	اروپا، ژاپن	ژاپن
کاربران نوعی	تاجران و صنعتگران	افراد جوان

شکل ۷-۵۴. مقایسه‌ای بین سیستمهای نسل اول WAP و I-Mode .

نسل دوم وب بیسیم

WAP 1.0 ، که بر اساس استانداردهای پذیرفته‌شدهٔ بین‌المللی طراحی شده بود، ابزاری جدی برای افراد پُرتحرک محسوب می‌شد. اما، متأسفانه شکست خورد. در مقابل، I-Mode بازیچه‌ای بود برای نوجوانان ژاپنی، که هیچ استانداردی هم پشت آن نبود. در میان تعجب همگان، I-Mode یک موفقیت تجاری بزرگ بود. بعد چه شد؟ نسل اول وب بیسیم درسهایی برای همه داشت. کنسرسیوم WAP فهمید که محتوا از همه چیز مهمتر است. اگر تعداد زیادی سایت وب که به زبان شما حرف بزنند نداشته باشید، باخته‌اید. از آن سو، NTT DoCoMo هم

فهمید که یک سیستم اختصاصی بسته (که فقط برای گوشی‌های کوچک و فرهنگ ژاپنی طراحی شده باشد) نمی‌تواند شانسی برای جهانی شدن داشته باشد. هر دو طرف هم فهمیدند که، برای متقاعد کردن سایتهای وب برای حرف زدن به فرمت شما، باید زبانی داشته باشید باز، محکم و مطابق با استانداردهای جهانی. در دنیای تجارت جنگ فرمتها چیز خوبی نیست.

هر دو سرویس در آستانه ورود به نسل دوم خود هستند. از آنجائیکه WAP 2.0 زودتر به بازار آمده، ما هم آنرا بررسی خواهیم کرد. WAP 1.0 چیزهای خوبی داشت، که WAP 2.0 آنها را حفظ کرده است. یکی اینکه، WAP می‌تواند روی شبکه‌های مختلف کار کند. نسل اول از شبکه‌های سوئیچینگ مداری استفاده می‌کرد، ولی همیشه می‌توانست با سوئیچینگ بسته‌ای هم کار کند. نسل دوم به احتمال زیاد از سوئیچینگ بسته‌ای (مثلاً، GPRS) استفاده خواهد کرد. دیگر اینکه، WAP از دستگاههای متنوعی (از تلفنهای همراه گرفته تا کامپیوترهای کتابی قوی) پشتیبانی می‌کرد، و هنوز هم می‌کند.

WAP 2.0 ویژگیهای جدیدی هم دارد، که مهمترین آنها عبارتند از:

۱. پشتیبانی از مدل پوش (دادن)، در کنار مدل قدیمی پول (گرفتن).
۲. یکپارچه کردن سرویسهای تلفنی در برنامه‌های کاربردی.
۳. پیام‌رسانی چندرسانه‌ای.
۴. داشتن ۲۶۴ علامت تصویری.
۵. ارتباط با وسایل ذخیره‌سازی.
۶. پشتیبانی از افزودنی‌های مرورگر.

مدل پول (pull) برای همه شناخته شده است: مشتری صفحه‌ای را درخواست می‌کند، و آنرا می‌گیرد. مدل پوش (push) یعنی فرستادن (دادن) اطلاعات به مشتری بدون اینکه درخواست کرده باشد، مانند ارسال پیوستهٔ قیمت سهام یا هشدارهای ترافیکی به کاربر.

مدتهاست که صدا و داده در حال یکی شدن هستند، و WAP 2.0 به طرق مختلف از آنها پشتیبانی می‌کند. یک نمونه که قبلاً به آن اشاره کردیم، ایجاد دفترچه تلفن تصویری با I-Mode است. WAP 2.0، علاوه بر ایمیل و تلفن، از پیام‌رسانی چندرسانه‌ای (multimedia messaging) هم پشتیبانی می‌کند.

محبوبیت فوق‌العادهٔ کاراکترهای اموجی در I-Mode، کنسرسیوم WAP را تشویق کرد که ۲۶۴ کاراکتر اموجی در WAP 2.0 بگنجاند. این کاراکترها در زمینه‌های متنوعی از قبیل حیوانات، وسایل خانگی، لباس، احساسات، غذا، بدن انسان، جنسیت، نقشه، موسیقی، گیاهان، ورزش، وقت، آب و هوا، ابزار، وسایط نقلیه، و اسلحه طراحی شده‌اند. جالبست بدانید که در استاندارد WAP 2.0 فقط نام این علائم تصویری تعریف شده، و اقدامی برای طراحی شکل آنها بعمل نیامده است، چون برخی از آنها (مانند بوسیدن و در آغوش گرفتن) در فرهنگهای مختلف معانی بسیار متفاوتی دارند. این مشکل در I-Mode وجود نداشت، چون فقط برای یک کشور طراحی شده بود.

پشتیبانی از وسایل ذخیره‌سازی بدان معنا نیست که هر تلفن WAP 2.0 یک هارد دیسک بزرگ خواهد داشت؛ Flash ROM نیز نوعی وسیلهٔ ذخیره‌سازی است. یک دوربین بیسیم WAP می‌تواند عکسهای گرفته شده را موقتاً روی یک Flash ROM ذخیره کند، تا زمان فرستادن بهترین آنها به اینترنت فرا برسد.

و بالاخره، مرورگرهای WAP 2.0 می‌توانند با استفاده از افزودنی‌ها (plug-in) قابلیتهای خود را توسعه دهند. در WAP 2.0 یک زبان اسکریپت‌نویسی نیز پیش‌بینی شده است.

تفاوتهای فنی مختلفی نیز بین WAP 1.0 و WAP 2.0 وجود دارد، که پشتهٔ پروتکل و زبان علامتگذاری از مهمترین آنهاست. WAP 2.0 همچنان به پشتیبانی از پشته پروتکل قدیمی شکل ۷-۴۸ ادامه می‌دهد، ولی از استاندارد اینترنت یعنی TCP و HTTP/1.1 نیز پشتیبانی می‌کند. پروتکل TCP در WAP 2.0 دارای چهار تفاوت

جزئی (ولی سازگار) با TCP استاندارد است: (۱) استفاده از پنجره‌های ثابت 64-KB ، (۲) نداشتن شروع آهسته، (۳) سقف ۱۵۰۰ بایتی برای MTU ، و (۴) تفاوت جزئی در الگوریتم ارسال مجدد (این تغییرات برای ساده‌تر شدن کُد پروتکل انجام شده‌اند). TLS یک پروتکل ایمنی لایهٔ انتقال (Transport Layer Security) است، که توسط IETF استاندارد شده است (برای توضیحات بیشتر به فصل ۸ مراجعه کنید). بسیاری از دستگاههای WAP 2.0 همزمان از هر دو پشته پروتکل پشتیبانی خواهند کرد (شکل ۷-۵۵ را ببینید).

WAP 1.0 protocol stack	WAP 2.0 protocol stack
XHTML	XHTML
WSP	HTTP
WTP	TLS
WTLS	TCP
WDP	IP
Bearer layer (لایه منتقل کننده)	Bearer layer (لایه منتقل کننده)

شکل ۷-۵۵. WAP 2.0 از دو پشته پروتکل پشتیبانی می‌کند.

تفاوت دیگر WAP 2.0 با WAP 1.0 در زبان علامتگذاری آنهاست. WAP 2.0 از XHTML ساده (که برای دستگاههای کوچک بیسیم طراحی شده) پشتیبانی می‌کند. از آنجائیکه NTT DoCoMo هم قبول کرده که از XHTML ساده پشتیبانی کند، با نوشتن صفحات خود به این زبان می‌توانید مطمئن باشید که آنها در اینترنت و دستگاههای بیسیم بخوبی دیده خواهند شد. این تصمیم به جنگ فرمتها (که رشد وب بیسیم را دچار وقفه کرده بود) خاتمه خواهد داد.

شاید بد نباشد چند کلمه‌ای هم دربارهٔ XHTML ساده بگوئیم. این زبان برای تلفنهای همراه، تلویزیونها، دستگاههای PDA ، ماشینهای فروش کالا، فراخوان‌ها، اتومبیل‌ها، کنسولهای بازی، و حتی ساعت در نظر گرفته شده است. بهمین دلیل در این ویرایش از شیوه‌نامه، اسکریپت، یا فریم خبری نیست، ولی اکثر برچسب‌های استاندارد وجود دارند. این برچسب‌ها به ۱۱ ماژول تقسیم شده‌اند، که برخی اجباری و برخی دیگر اختیاری‌اند (تمام این ماژولها در XML تعریف شده‌اند). این ماژولها را، همراه با مثالهایی از هر کدام، در شکل ۷-۵۶ ملاحظه می‌کنید. برای کسب اطلاعات بیشتر دربارهٔ این برچسب‌ها می‌توانید به *www.w3.org* مراجعه کنید.

علیرغم توافق WAP و I-Mode بر سر استفاده از XHTML ساده، رقیب جدیدی آنها را تهدید می‌کند: 802.11 . نسل دوم وب بیسیم با سرعت 384 kbps کار می‌کند، که از 9600 bps نسل اول خیلی بهتر است، ولی در مقایسه با سرعت 11 Mbps یا 54 Mbps دستگاههای 802.11 رنگ می‌بازد. البته 802.11 همه جا حضور ندارد، ولی با تصمیم صاحبان رستورانها، هتلها، فروشگاهها، شرکتها، فرودگاهها، ایستگاههای اتوبوس، موزه‌ها، دانشگاهها، بیمارستانها، و بسیاری جاهای دیگر برای نصب ایستگاههای مرکزی 802.11 و دادن امکان دسترسی اینترنت به کارمندان و مشتریان خود، پوشش کافی در مناطق شهری بوجود خواهد آمد، و آن وقت کافیست به نزدیکترین کافه تریا بروید تا بتوانید ضمن خوردن یک فنجان قهوه، ایمیلهای خود را هم چک کنید. دور نیست روزی که مغازه‌ها (کنار آرم کارتهای اعتباری قابل قبول) برای جلب مشتری بیشتر آرم 802.11 را در ویترین یا کنار در ورودی خود قرار دهند، و مردم هم (مثل صحرانشینان که بدنبال آب بیابانها را زیر پا می‌گذاشتند) بدنبال 802.11 از این خیابان به آن خیابان سرگردان شوند.

برچسب های نمونه	کارکرد	الزامی؟	ماژول
body, head, html, title	ساختار سند	بلی	ساختاری
br, code, dfn, em, h*n*, kbd, p, strong	اطلاعات	بلی	متن
a	ابرلینک	بلی	ابر متن
dl, dt, dd, ol, ul, li	لیست	بلی	لیست
form, input, label, option, textarea	فرم	خیر	فرم
caption, table, td, th, tr	جدول	خیر	جدول
img	تصویر	خیر	تصویر
object, param	اپلت، نقشه و غیره	خیر	شیء
meta	اطلاعات اضافی	خیر	اطلاعات اضافی
link	شبیه <a>	خیر	لینک
base	نقطه شروع URL	خیر	مبنا

شکل ۷-۵۶. ماژول‌ها و برچسب‌های اصلی XHTML .

شاید رستورانها و کافه‌تریاها خیلی زود برای نصب ایستگاههای مرکزی 802.11 دست بکار شوند، ولی بنظر نمی‌رسد کشاورزان به این زودی‌ها چنین کاری بکنند، بهمین دلیل پوشش یکپارچهٔ 802.11 به مناطق شهری (و حداکثر حومهٔ آنها) محدود خواهد ماند (بیاد بیاورید که بُرد 802.11 در بهترین حالت چند صد متر بیشتر نیست). شاید در آینده دستگاههایی به بازار بیایند که در صورت یافتن سیگنال 802.11 از آن استفاده کنند، و وقتی به جایی رسیدند که این پوشش وجود نداشت، بطور خودکار به WAP سوئیچ کنند.

۷-٤ چندرسانه‌ای

وب بیسیم یکی از تکنولوژیهای جدید و مهیج است، ولی تنها تکنولوژی نیست. برای خیلی‌ها چندرسانه‌ای (multimedia) جام مقدس شبکه است. وقتی اسم چندرسانه‌ای می‌آید، همهٔ چشمها خیره می‌شود (چون برای هر کس چیزی دارد). از آنجائیکه چندرسانه‌ای به پهنای باند زیادی نیاز دارد، فعلاً فقط روی شبکه‌های ثابت کار می‌کند، و برای دیدن آن روی لینکهای بیسیم باید چند سال دیگر منتظر ماند.

از نظر لغوی، چندرسانه‌ای یعنی دو یا چند رسانه. حتی ناشر همین کتاب حاضر هم می‌تواند برای آن بعنوان کتابی چندرسانه‌ای تبلیغ کند، چون هم متن دارد هم تصویر (شکل). با این حال، وقتی اغلب مردم این اصطلاح را بکار می‌برند، منظورشان ترکیبی از چند **رسانهٔ پیوسته** (continuous media) – رسانه‌هایی که می‌توان آنها در یک فاصلهٔ زمانی مشخص، بهمراه نوعی تعامل با کاربر، پخش کرد – است. در عمل، چندرسانه‌ای بیشتر به ترکیبی از صدا و ویدئو (تصاویر متحرک) اطلاق می‌شود.

با این همه، بسیاری از مردم به صدای تنها (مثلاً، تلفن یا رادیوی اینترنتی) هم چندرسانه‌ای می‌گویند، که پیداست چنین نیست. نام **رسانهٔ جویباری** (streaming media) برای این قبیل رسانه‌ها مناسبتر است، ولی اجازه دهید ما هم با افکار عمومی همراهی کنیم و آنرا همان چندرسانه‌ای بخوانیم. در قسمتهای آینده خواهید دید که کامپیوترها چگونه صدا و ویدئو را پردازش می‌کنند، چگونه آنها را فشرده می‌کنند، و چگونه می‌توان از این تکنولوژیها استفاده کرد. برای یک بحث مفصل (سه جلدی) دربارهٔ شبکه‌های چندرسانه‌ای کتابهای (Steinmetz and Nahrstedt, 2002; Steinmetz and Nahrstedt, 2003; and Steinmetz and Nahrstedt, 2003b) را ببینید.

۷-۴-۱ مقدمه‌ای بر صدای دیجیتال

صدا یک موج فشاری یک بُعدی است. وقتی این موج وارد گوش می‌شود، پردهٔ گوش را به ارتعاش در آورده و باعث می‌شود تا استخوانهای ریز گوش میانی هم به ارتعاش در آیند. ارتعاش استخوانهای گوش میانی باعث ایجاد سیگنالهای عصبی در گوش داخلی شده، و این سیگنال به مغز می‌رود. درک ما از صدا توسط مغز (و با تفسیر این سیگنالهای عصبی) انجام می‌شود. به همین ترتیب، وقتی موج صوتی به میکروفون برخورد می‌کند، میکروفون آنرا به یک سیگنال الکتریکی (که دامنهٔ آن تابعی از زمان است) تبدیل می‌کند. پردازش، ذخیره‌سازی، انتقال و بازتولید این سیگنالها یکی از مهمترین زمینه‌های تحقیقاتی سیستمهای چندرسانه‌ای است.

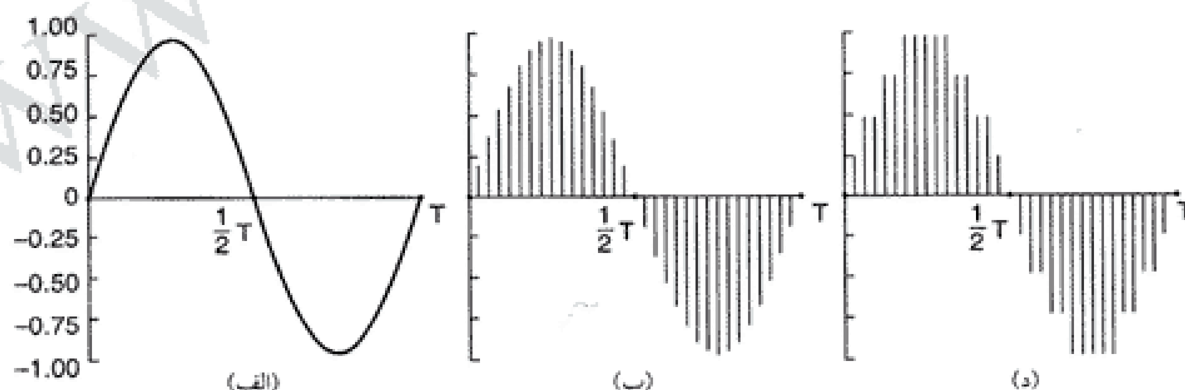
محدودهٔ فرکانسی گوش انسان بین 20 Hz تا 20,000 Hz است. برخی از حیوانات (بویژه سگ و خفاش) می‌توانند فرکانسهای بالاتر را نیز بشنوند. توانایی گوش در شنیدن اصوات لگاریتمی است، بنابراین نسبت دو صوت با قدرتهای A و B بصورت dB (دسی‌بل) با فرمول زیر بیان می‌شود:

$$dB = 10\log_{10}(A/B)$$

اگر حد پائین شنوایی (فشاری معادل $0.0003\ dyne/cm^2$) در فرکانس 1-kHz سینوسی را 0 dB تعریف کنیم، صحبت معمولی حدود 5 dB ، و آستانهٔ درد معادل 120 dB است (یعنی تفاوتی در حد ۱ میلیون برابر).

حساسیت گوش به صداهای گذرا (در حد چند میلی‌ثانیه) بسیار شگفت‌آور است (حتی چشم نیز نورهایی در این فاصلهٔ کوتاه را تشخیص نمی‌دهند). در نتیجه، لرزش صدا (حتی برای چند میلی‌ثانیه) می‌تواند در کیفیت انتقال چندرسانه‌ای اثر بگذارد، در حالیکه چنین لرزشهایی روی کیفیت تصاویر تقریباً بی‌تأثیر است.

برای تبدیل صدا به سیگنال دیجیتال می‌توان از یک ADC (مبدل آنالوگ‌دیجیتال – Analog Digital Converter) استفاده کرد. مبدل ADC ولتاژ الکتریکی را بعنوان ورودی گرفته، و یک خروجی دودویی (باینری) تولید می‌کند. شکل ۷-۵۷ (الف) یک موج سینوسی را نشان می‌دهد. برای نمایش دیجیتالی این سیگنال، می‌توانیم در فواصل ΔT ثانیه از آن نمونه‌برداری کنیم (شکل ۷-۵۷ ب). اگر موج صوتی سینوسی خالص نباشد ولی بتوان آنرا بصورت مجموع خطی چند موج سینوسی (با بیشترین فرکانس f) نمایش داد، قضیهٔ نایکوئیست (فصل ۲) می‌گوید که فرکانس $2f$ برای نمونه‌برداری آن کافیست. نمونه‌برداری با فرکانسهای بالاتر اطلاعات بیشتری بدست نمی‌دهد، چون فرکانسهایی که این نمونه‌برداری می‌تواند آشکار کند، در موج اصلی وجود ندارند.



شکل ۷-۵۷. (الف) یک موج سینوسی. (ب) نمونه‌برداری از موج سینوسی. (ج) کوانتیزه کردن نمونه‌ها بصورت ۴ بیتی.

نمونه‌های دیجیتالی هرگز دقیق و کامل نیستند. در مثال شکل ۷-۵۷ (ج) فقط ۹ مقدار مجاز (از 1.00- تا 1.00+ باگامهای 0.25) وجود دارد (و همانطور که می‌دانید، برای نمایش ۹ مقدار به ۴ بیت دودویی نیاز داریم). با

نمونه‌های ۸بیتی می‌توان ۲۵۶ مقدار مختلف را ثبت کرد؛ و با نمونه‌های ۱۶ بیتی می‌توانیم ۶۵،۵۳۶ مقدار مجزا داشته باشیم. خطایی که در اثر محدود بودن تعداد بیتها برای نمایش هر نمونه بوجود می‌آید، به **نویزکوانتیزه‌کردن** (quantization noise) معروف است. اگر این خطا زیاد باشد، گوش می‌تواند آنرا تشخیص دهد.

تلفن و دیسکهای صوتی دو نمونهٔ آشنا از کوانتیزه کردن صوت هستند. مدولاسیون کُد پالس (PCM) ، که در سیستمهای تلفن بکار می‌رود، از نمونه‌های ۸بیتی استفاده می‌کند و در هر ثانیه ۸۰۰۰ نمونه برمی‌دارد. در آمریکای شمالی و ژاپن، ۷ بیت برای داده و ۱ بیت برای کنترل بکار می‌رود، در حالیکه در اروپا از هر ۸ بیت برای داده استفاده می‌شود. نرخ داده در این سیستمها بترتیب 56,000 bps و 64,000 bps است. با نرخ نمونه‌برداری 8000 samples/sec فرکانسهای بالاتر از 4 kHz از دست می‌روند.

نرخ نمونه‌برداری در دیسکهای صوتی 44,100 samples/sec است، که برای آشکارسازی فرکانسهایی تا 22,050 Hz کافیست (که برای آدمهای موزیک دوست خوب است، ولی سگ‌ها بهیچوجه از آن راضی نخواهند بود!). در این دیسکها، نمونه‌ها ۱۶ بیتی (با توزیع خطی روی محدودهٔ دامنه) هستند. توجه کنید که ۱۶ بیت فقط برای نمایش ۶۵،۵۳۶ مقدار کافیست، در حالیکه بین پائینترین و بالاترین حد شنوایی انسان حداقل ۱،۰۰۰،۰۰۰ گام قابل شنیدن وجود دارد؛ بهمین دلیل، حتی در این دیسکها هم مقداری نویز کوانتیزه کردن وجود دارد. با 44,100 نمونهٔ ۱۶بیتی در هر ثانیه، دیسکهای صوتی به پهنای باند 705.6 kbps برای اصوات مونو، و 1.411 Mbps برای اصوات استریو نیاز دارند. همانطور که می‌بینید، برای انتقال صوت غیرفشرده با کیفیت CD به یک کانال کامل T1 نیاز داریم (در مورد ویدئو که وضع از این هم بدتر است - قسمت بعد را ببینید).

کامپیوترها می‌توانند صدای دیجیتال را براحتی پردازش کنند. امروزه برنامه‌های بسیاری برای ضبط، پخش، ادیت، میکس و ذخیره کردن امواج صوتی در بازار یافت می‌شود، و تقریباً تمام حرفه‌ایها برای ضبط و ادیت کردن صوت از سیستمهای دیجیتال استفاده می‌کنند.

صحبت یکی دیگر از زمینه‌های مهم در صدای دیجیتال است. صحبت انسانها معمولاً در محدودهٔ 600 Hz تا 6000 Hz است. حروف صدادار و بی‌صدا هر کدام ویژگیهای خاص خود دارند. حروف صدادار وقتی ایجاد می‌شوند که مجرای صوتی باز است و تشدید صورت می‌گیرد، و فرکانس آن به اندازه و شکل حنجره، و محل قرار گرفتن زبان و آروارهٔ فرد بستگی دارد. اصوات صدادار تناوبی در حدود 30 msec دارند. حروف بی‌صدا زمانی ایجاد می‌شوند که مجرای صوتی تقریباً مسدود است، و معمولاً بدون تناوب و شکل خاصی هستند.

در برخی از سیستمهای تولید صحبت (بجای ترکیب شکل‌موج واژه‌ها) از مدلهای ساده شدهٔ مجرای صوتی (با پارامترهای محدودتری از قبیل اندازه و شکل حفره‌های صوتی) استفاده می‌شود. طرز کار این سیستمها در حیطهٔ بحث کتاب حاضر نیست.

۷-۴-۲ فشرده‌سازی صدا

همانطور که دیدید برای انتقال صدای استریو با کیفیت CD به پهنای باند 1.411 Mbps نیاز داریم، پس پیداست که برای انتقال این رسانه روی اینترنت باید آنرا تا حد زیادی فشرده کنیم. برای این کار الگوریتمهای فشرده‌سازی مختلفی توسعه داده شده‌اند، که شاید محبوبترین آنها صدای MPEG باشد. این الگوریتم دارای سه لایه (نوع) مختلف است، که لایهٔ سوم (MPEG audio layer 3 - MP3) از همه قویتر و معروفتر است. حجم زیادی از موسیقی با این فرمت روی اینترنت وجود دارد، که البته همهٔ آنها قانونی نیستند، و همین منجر به دعواهای قانونی بسیاری بین متخلفان و صاحبان این آثار شده است. فرمت MP3 در واقع بخش صوتی استاندارد فشرده‌سازی ویدئویی MPEG است، که در قسمتهای آینده دربارهٔ آن هم صحبت خواهیم کرد.

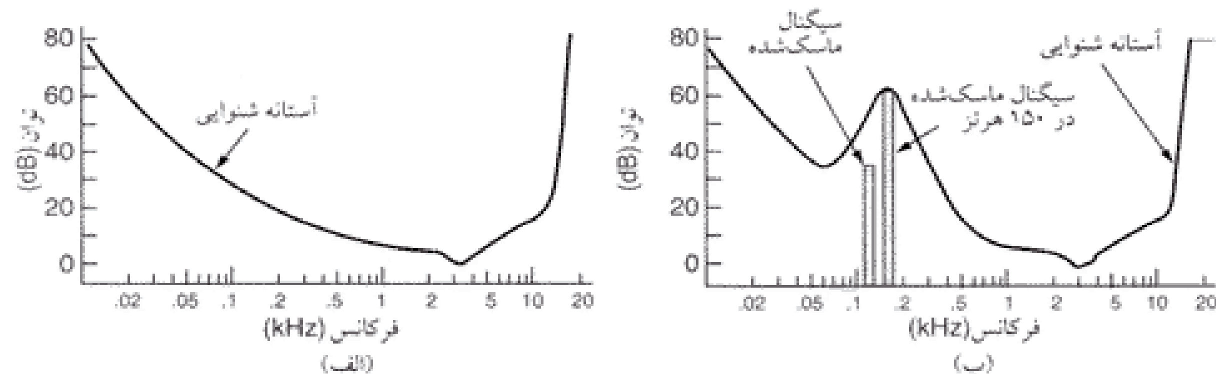
فشرده‌سازی صدا را به دو روش می‌توان انجام داد. در **کُدگذاری شکل‌موج** (waveform coding) سیگنال

صوتی بصورت ریاضی و با استفاده از تبدیل فوریه، به فرکانسهای تشکیل‌دهندهٔ آن تجزیه می‌شود. در شکل ۲-۱ (الف) نمونه‌ای از یک موج و دامنه‌های فوریهٔ آنرا می‌بینید. سپس دامنهٔ هر مؤلفه به ساده‌ترین شکل ممکن کُد می‌شود. هدف از این کار بازسازی شکل‌موج با حداقل بیت‌های ممکن است.

روش دیگر، **کُدگذاری‌ادراکی** (perceptual coding) ، سیگنال ورودی را با استفاده از نقائص سیستم صوتی انسان بگونه‌ای کُد می‌کند که شنونده متوجه تفاوت آنها نشود (حتی اگر این سیگنالها در اسیلوسکوپ بهیچوجه یکسان دیده نشوند). کُدگذاری ادراکی بر اساس تحقیقات **روان‌شنوایی** (psychoacoustics) - دانش درک انسانها از صوت - بنا نهاده شده است. فرمت MP3 از کُدگذاری ادراکی استفاده می‌کند.

نکتهٔ کلیدی در کُدگذاری ادراکی این است که برخی از اصوات می‌توانند صداهای دیگر را **پوشانند**. فرض کنید در یک روز تابستانی در هوای آزاد مشغول اجرای یک کنسرت فلوت هستید. ناگهان وسط کنسرت شما چند متر آن طرفتر کارگران شهرداری چکشهای بادی خود را روشن کرده و مشغول کندن آسفالت خیابان می‌شوند. در این حال دیگر هیچکس نمی‌تواند صدای فلوت شما را بشنود، چون صدای چکشهای بادی نوای فلوت را پوشانده است. اگر بخواهید همین کنسرت را به جای دیگر مخابره کنید، فقط کافیست صدای چکشهای بادی را ارسال کنید، چون به هر حال کسی صدای فلوت را نخواهد شنید. به این پدیده - پوشانده شدن صدای ملایم یک باند فرکانسی توسط صدای خشن یک باند فرکانسی دیگر - **ماسک‌فرکانسی** (frequency masking) می‌گویند. در حقیقت، حتی بعد از خاموش شدن چکشهای بادی باز هم تا مدتی شنوندگان قادر به شنیدن صدای فلوت نخواهند بود، چون گوش (برای جلوگیری از صدمه دیدن سیستم شنوایی) بهرهٔ شنوایی خود را با شروع صدای چکشها پائین می‌آورد، و بازگشت آن به حالت قبل (بعد از خاموش شدن چکشها) مدتی طول خواهد کشید. به این پدیده هم **ماسک‌موقتی** (temporal masking) گفته می‌شود.

برای دیدن تأثیر کمی این پدیده‌ها، یک آزمایش ترتیب می‌دهیم. در یک اتاق کاملاً ساکت، فردی گوشی‌های متصل به کارت صوتی کامپیوتر را به گوش می‌گذارد. کامپیوتر یک موج سینوسی کم قدرت با فرکانس 100 Hz تولید می‌کند، و بتدریج قدرت آنرا بالا می‌برد. به این فرد گفته‌ایم که به محض شنیدن صدا کلیدی را بزند. در این لحظه کامپیوتر قدرت صدا را ثبت کرده، و همین کار را با فرکانسهای 200 Hz ، 300 Hz و ... (تا رسیدن به آستانهٔ شنوای انسان) تکرار می‌کند. با تکرار این آزمایش برای افراد مختلف و محاسبهٔ متوسط قدرت لازم برای شنیده شدن صدا در هر فرکانس، نمودار لگاریتمی شکل ۷-۵۸ (الف) را رسم کرده‌ایم. این نمودار بروشنی نشان می‌دهد که هیچ نیازی به کُد کردن فرکانسهایی که قدرت آنها زیر آستانهٔ شنوایی انسان باشد، نیست. برای مثال، اگر قدرت یک سیگنال 100 Hz فقط 20 dB باشد، می‌توان آنرا حذف کرد بدون اینکه تأثیری روی کیفیت خروجی بگذارد، چون طبق شکل ۷-۵۸ (الف) این سیگنال زیر آستانهٔ شنوایی انسانها قرار دارد.



شکل ۷-۵۸. (الف) نمودار آستانهٔ شنوایی بر حسب فرکانس. (ب) اثر ماسک فرکانسی.

اکنون آزمایش دیگری می‌کنیم. در اینجا هم کامپیوتر همان تست قبل را اجرا می‌کند، ولی این بار یک موج سینوسی ثابت با فرکانس (مثلاً 150 Hz) در زمینهٔ فرکانس تست پخش می‌شود. همانطور که به روشنی می‌توان دید (شکل ۷–۵۸ ب)، آستانهٔ شنوایی فرکانسهای نزدیک 150 Hz بالاتر رفته است.

نتیجهٔ این آزمایش آن است که، با دنبال کردن فرکانسهای نزدیک به هم می‌توان سیگنالهای ضعیفتر را حذف کرده، و در تعداد بیت‌های موردنیاز صرفه‌جویی کرد. شکل ۷–۵۸ (ب)نشان می‌دهد که حتی اگر فرکانسهای 125-Hz را بکلی حذف کنیم، هیچ گوشی متوجه تفاوت آن نخواهد شد. حتی اگر در جایی سیگنال قویتر متوقف شود، باز هم می‌توانیم تا مدتی به حذف سیگنال ضعیفتر ادامه دهیم، چون برگشت گوش به حالت نرمال کمی طول می‌کشد (اثر ماسک موقتی). فرمت MP3 نیز در اساس چیزی نیست جز تبدیل فوریهٔ صدا به سیگنالهای سینوسی و حذف تمام سیگنالهایی که ماسک شده‌اند.

با این زمینهٔ تئوریک، اکنون می‌توانیم نشان دهیم که کُدگذاری چگونه انجام می‌شود. برای فشرده‌سازی صدا، از شکل موج ورودی با نرخهای 32 kHz ، 44.1 kHz ، یا 48 kHz نمونه‌برداری می‌شود. نمونه‌برداری را می‌توان روی یک یا دو کانال به طرق زیر انجام داد:

۱. تک‌صدایی (یک استریم ورودی).
۲. تک‌صدایی دوبل (مثلاً، حاشیهٔ صوتی انگلیسی و ژاپنی).
۳. استریوی منفصل (فشرده‌سازی جداگانهٔ هر کانال).
۴. استریوی متصل (استفادهٔ کامل از افزونگی بین‌کانالها).

ابتدا، نرخ بیت خروجی انتخاب می‌شود. با الگوریتم MP3 می‌توان یک CD استریوی راک اند رول را تا 96 kbps فشرده کرد، بطوریکه حتی دوآتشه‌ترین طرفداران این موسیقی هم متوجه افت کیفیت نشوند. برای یک کنسرت پیانو به حداقل 128 kbps نیاز داریم، چون نسبت سیگنال–به–نویز موسیقی راک اند رول بسیار بالاتر از پیانو است. اگر نرخ پائینتری انتخاب کنیم، کیفیت صدا قدری افت خواهد کرد.

سپس، نمونه‌برداری در گروههای ۱۱۵۲ تایی (تقریباً 26 msec) انجام می‌شود. هر گروه با عبور از ۳۲ فیلتر دیجیتال به ۳۲ باند فرکانسی تفکیک می‌شود. همزمان، موج ورودی به یک مدل روان‌شنوایی داده می‌شود تا فرکانسهای ماسک شده مشخص شوند. پس از آن ۳۲ باند فرکانسی باز هم تبدیل می‌شوند، تا طیف فرکانسی دقیقتری بدست آید.

در مرحلهٔ بعد، تعداد بیت‌های موجود بین این باندها تقسیم می‌شود (با این معیار که به توانهای طیفی ماسک‌نشده بیت‌های بیشتری برسد، باندهای ماسک‌نشدهٔ کم‌قدرت‌تر بیت‌های کمتری بگیرند، و باندهای ماسک‌شده اصلاً چیزی نگیرند). در پایان هم، بیت‌های حاصله با استفاده از روش هافمن – اختصاص کُدهای کوتاهتر به اعدادی که بیشتر تکرار شده‌اند، و اختصاص کُدهای بلندتر به اعداد کمتر تکرار شده – کُد می‌شوند.

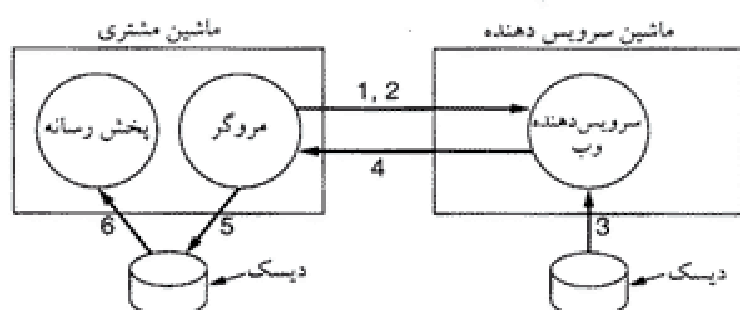
در واقع، قصهٔ فشرده‌سازی مفصلتر از اینهاست. تکنیکهای مختلفی برای کاهش نویز، یکنواخت‌سازی، و بهره‌گیری از افزونگی بین‌کانالها وجود دارد، که از حوزهٔ بحث این کتاب خارج هستند. برای کسب اطلاعات بیشتر در زمینهٔ ریاضیات فشرده‌سازی صدا به (Pan, 1995) مراجعه کنید.

۷–٤–۳ صدای جویباری

در این قسمت سه تا از کاربردهای صدای دیجیتال را مورد بررسی قرار می‌دهیم. اولین آنها **صدای جویباری** (streaming audio) است: شنیدن صدا بصورت پیوسته روی اینترنت. در قسمتهای بعد با رادیوی اینترنتی و صدا روی IP (یا همان تلفن اینترنتی) آشنا خواهید شد.

اینترنت پُر است از سایتهای موسیقی، که کاربر می‌تواند با کلیک کردن روی لینک هر ترانه به آن گوش کند.

بعضی از این سایتها مجانی هستند (گروههای تازه‌کاری که بدنبال شهرت‌اند)، و برای بعضی دیگر باید پول داد (البته روی این سایتها هم نمونه‌های مجانی یافت می‌شود). ساده‌ترین روش پخش موسیقی را در شکل ۷–۵۹ ملاحظه می‌کنید.



شکل ۷–۵۹. یک روش ساده برای پخش موسیقی در صفحات وب.

کار با کلیک کردن روی لینک موسیقی دلخواه شروع می‌شود، و در اینجا مرورگر وارد صحنه می‌شود. در مرحلهٔ ۱ مرورگر یک اتصال TCP به سرویس‌دهندهٔ وب هدف برقرار می‌کند، و در مرحلهٔ ۲ با فرمان *GET* فایل موسیقی را از آن درخواست می‌کند. سپس (مراحل ۳ و ۴)، سرویس‌دهنده فایل خواسته شده را (که می‌تواند با فرمت MP3 یا هر فرمت دیگری باشد) از روی دیسک خوانده و برای مرورگر می‌فرستد. اگر این فایل از حجم حافظهٔ سرویس‌دهنده بزرگتر باشد، آنرا بصورت قطعه قطعه خواهد فرستاد.

مرورگر با استفاده از نوع MIME فایل دریافت‌شده (مثلاً، *audio/mp3*)، تشخیص می‌دهد که چگونه باید آنرا پخش کند. مرورگرها معمولاً برای این کار از برنامه‌های کمکی مانند RealOne Player ، Windows Media Player ، یا Winamp کمک می‌گیرند. از آنجائیکه متداولترین روش ارتباط با برنامه‌های کمکی نوشتن فایل روی محلی موقتی است، مرورگر ابتدا فایل را روی دیسک ذخیره می‌کند (مرحلهٔ ۵). سپس، برنامهٔ پخش موسیقی را اجرا کرده و نام فایل را به آن می‌دهد. با این کار برنامهٔ پخش موسیقی شروع به خواندن فایل از روی دیسک، و پخش آن می‌کند (مرحلهٔ ۶).

این روش هیچ اشکالی ندارد و در عمل کار هم می‌کند، ولی مشکل اینجاست که تمام فایل باید قبل از شروع پخش خوانده شود. اگر یک فایل موسیقی 4 MB باشد (اندازه‌ای معمولی برای فایلهای MP3)، و کاربر با مودم 56 kbps به اینترنت متصل شده باشد، قبل از شروع پخش موسیقی باید ۱۰ دقیقه انتظار بکشد. اغلب موسیقی‌دوستان تحمل چنین انتظاری را ندارند.

برخی از سایتهای وب برای حل این مشکل (بدون آنکه روش دسترسی به موسیقی را تغییر دهند) از روش ذیل استفاده می‌کنند. در این روش، لینک صفحهٔ وب مستقیماً به فایل موسیقی اشاره نمی‌کند، بلکه یک متافایل (metafile – فایل بسیار کوچکی که فقط نام فایل موسیقی در آن است) را مشخص می‌کند. یک متافایل چنین شکلی دارد:

```
rtsp://joes-audio-server/song-0025.mp3
```

وقتی مرورگر این فایل یک‌خطی را دریافت کرد، طبق معمول آنرا روی دیسک نوشته و پس از اجرای برنامهٔ کمکی پخش موسیقی، نام فایل مزبور را به آن می‌دهد. برنامهٔ پخش بعد از خواندن فایل متوجه می‌شود که این یک URL است، بنابراین به سرویس‌دهندهٔ *joes-audio-server* متصل شده و آهنگ مشخص شده را درخواست می‌کند. توجه کنید که در اینجا دیگر مرورگر نقشی ندارد.

در اغلب موارد، سرویس‌دهندهٔ مشخص شده در متافایل همان سرویس‌دهندهٔ وب نیست، و در واقع حتی یک

سرویس‌دهندهٔ HTTP هم نیست، بلکه نوع خاصی از **سرویس‌دهندهٔ رسانه** (media server) است. در مثال بالا، سرویس‌دهندهٔ رسانه از پروتکل RTSP (**پروتکل جویباری بی‌درنگ** – Real Time Streaming Protocol) استفاده می‌کند (به نام پروتکل *rtsp* در ابتدای URL دقت کنید). این پروتکل در RFC 2326 تعریف شده است.

برنامهٔ پخش چهار وظیفهٔ اصلی دارد:

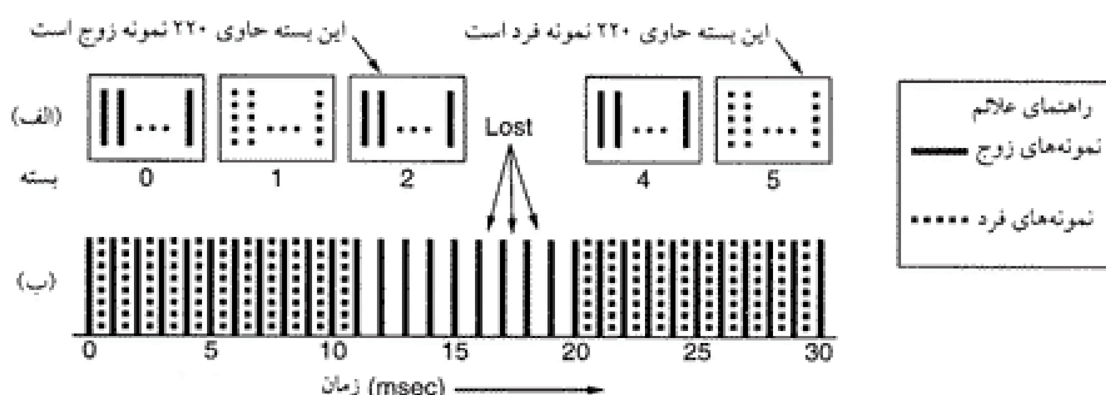
۱. مدیریت واسط کاربر.
۲. مقابله با خطاهای انتقال.
۳. باز کردن (از حالت فشرده خارج کردن) موسیقی.
۴. حذف لرزش‌ها.

اغلب برنامه‌های پخش امروزی دارای ظاهری بسیار جالب و شکیل هستند، و از دستگاههای پخش استریوی مدل بالا (با تمام دکمه، عقربه‌ها، و چراغهای رنگارنگ) تقلید می‌کنند. بعضی از آنها حتی اجازه می‌دهند تا شکل ظاهری برنامه را (که به پوست – skin – معروف است) به سلیقهٔ خود تغییر دهید. این یکی از وظایف برنامهٔ پخش در رابطه با واسط کاربر است.

وظیفهٔ دوم برنامهٔ پخش مقابله با خطاهای انتقال است. برنامه‌های پخش بی‌درنگ بندرت از TCP برای انتقال موسیقی استفاده می‌کنند، چون مدیریت خطا در TCP می‌تواند باعث ایجاد وقفه‌های آزاردهنده در موسیقی شود. پروتکل مرسوم برای انتقال موسیقی پروتکل RTP است، که آنرا در فصل ۶ دیدید. مانند اغلب پروتکلهای بی‌درنگ، RTP هم لایه‌ایست بر فراز UDP، بنابراین گم شدن بسته‌ها کاملاً محتمل است.

در برخی از موارد، برای بهبود مدیریت خطا از روشهای **یک‌درمیانی** (interleaving) استفاده می‌شود. برای مثال، هر بسته می‌تواند حاوی ۲۲۰ نمونهٔ استریو (یک زوج ۱۶ بیتی) معادل 5 msec موسیقی باشد، ولی پروتکل انتقال بجای آن ابتدا 10 msec از نمونه‌های فرد را در یک بسته، و بدنبال آن 10 msec از نمونه‌های زوج را در بستهٔ بعدی می‌فرستد. در این روش اگر یک بسته گم شود، شکاف 5 msec در موسیقی بوجود نمی‌آید، و برنامهٔ پخش می‌تواند با استفاده از تکنیکهای درون‌یابی زوجهای گم شده را تخمین بزند.

در شکل ۷-۶۰ تکنیک یک‌درمیانی را ملاحظه می‌کنید. در اینجا هر بسته شامل نمونه‌های فرد یا زوج فاصلهٔ زمانی 10 msec است. در نتیجه، از دست رفتن بستهٔ ۳ باعث ایجاد شکاف در موسیقی نمی‌شود، بلکه فقط کیفیت پخش بصورت موقتی و زودگذر افت خواهد کرد. برنامهٔ پخش برای حفظ پیوستگی موسیقی، با استفاده از تکنیکهای ریاضی درون‌یابی (interpolating) نمونه‌های گم شده را (از روی نمونه‌های قبلی و بعدی) تخمین

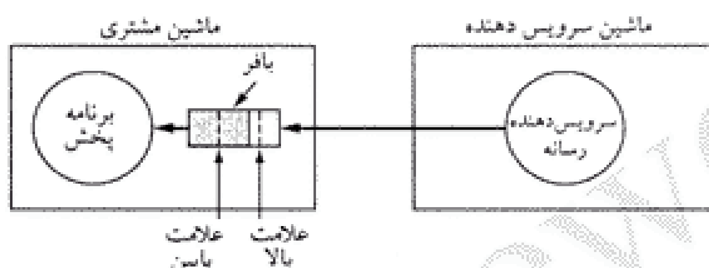


شکل ۷-۶۰. وقتی نمونه‌ها بصورت یک‌درمیان فرستاده شوند، گم شدن یک بسته بجای ایجاد شکاف در موسیقی فقط باعث کاهش موقتی کیفیت آن خواهد شد.

می‌زند. البته این روش خاص فقط با نمونه‌های فشرده‌نشده کار می‌کند، ولی بخوبی نشان می‌دهد که روشهای هوشمندانه چگونه می‌توانند به بهبود کیفیت پخش کمک کنند. (در RFC 3119 برای صدای فشرده‌شده نیز روشی ارائه شده است.)

وظیفهٔ سوم برنامهٔ پخش باز کردن (decompress) موسیقی است. با آن که این تکنیک نسبتاً ساده است، اما به محاسبات زیادی نیاز دارد.

وظیفهٔ چهارم برنامهٔ پخش، حذف لرزش (jitter) ، جزء آزاردهنده‌ترین وظایف همهٔ سیستمهای بی‌درنگ است. تمام سیستمهای صدای جویباری قبل از شروع پخش، ۱۰ الی ۱۵ ثانیه موسیقی را بافر می‌کنند (شکل ۷-۶۱ را ببینید). بطور ایده‌آل، سرویس‌دهنده این بافر را با همان سرعتی که برنامهٔ پخش از آن می‌خواند، پُر می‌کند. اما در واقعیت ممکنست چنین چیزی عملی نباشد، پس به نوعی حلقهٔ فیدبک نیاز داریم.



شکل ۷-۶۱. برنامهٔ پخش، بجای پخش مستقیم موسیقی از شبکه، ورودی سرویس‌دهندهٔ رسانه را بافر کرده، و موسیقی را از این بافر اجرا می‌کند.

برای پُر نگه داشتن بافر از دو روش می‌توان استفاده کرد. در روش **سرویس‌دهندهٔ پول** (pull server) ، تا زمانی که بافر جا دارد، برنامهٔ پخش به درخواست از سرویس‌دهندهٔ رسانه برای ارسال بسته‌های بعدی ادامه می‌دهد. هدف برنامهٔ پخش آن است که بافر را تا حد امکان پُر نگه دارد. عیب بزرگ این روش ارسال درخواستهای غیرضروری از طرف برنامهٔ پخش است. سرویس‌دهنده می‌داند که باید تمام فایل را بفرستد، پس درخواست پخش‌کننده دیگر برای چیست؟ به همین دلیل از روش فوق بندرت استفاده می‌شود.

در روش دیگر، **سرویس‌دهندهٔ پوش** (push server) ، برنامهٔ پخش فقط یک بار درخواست *PLAY* را به سرویس‌دهندهٔ رسانه می‌فرستد، و این سرویس‌دهنده به وظیفهٔ خود (فرستادن پیوستهٔ داده‌ها) عمل می‌کند. در اینجا دو احتمال پیش می‌آید: سرویس‌دهندهٔ رسانه با سرعت معمولی پخش هماهنگ یا از آن سریعتر است. در هر دو حالت، قبل از شروع پخش مقداری داده بافر می‌شود. اگر سرعت سرویس‌دهنده همپای پخش باشد، داده از آن وارد بافر شده و از سمت دیگر توسط برنامهٔ پخش خوانده می‌شود. اگر همه چیز خوب پیش رود، مقدار دادهٔ درون بافر همیشه ثابت می‌ماند. این ساده‌ترین حالت است، چون لازم نیست هیچ پیام کنترلی (در هر دو جهت) فرستاده شود.

حالت دیگر اینست که سرویس‌دهندهٔ پوش داده را با سرعتی بیشتر از آنچه برنامهٔ پخش می‌خواند، بفرستد. در این وضعیت سرویس‌دهنده همیشه می‌تواند خود را به برنامهٔ پخش برساند، حتی اگر گاهی (بدلایلی) از آن عقب بماند. اما این حالت (عجلهٔ سرویس‌دهنده برای عقب نماندن)، می‌تواند منجر به رونویسی بافر (buffer overrun) – نوشتن روی داده‌هایی که هنوز خوانده نشده‌اند – شود.

برنامهٔ پخش برای جلوگیری از این وضعیت می‌تواند دو علامت **حدپائین** و **حدبالا** در بافر تعریف کند. سرویس‌دهنده آنقدر داده می‌فرستد که بافر تا علامت حد بالا پُر شود؛ در اینجا پخش‌کننده به سرویس‌دهنده پیام

می‌دهد که فرستادن را متوقف کند. از آنجائیکه سرویس‌دهنده قبل از دریافت پیام توقف همچنان به ارسال داده ادامه می‌دهد، فاصلهٔ علامت حد بالای بافر تا ظرفیت نهایی آن باید از حاصلضرب پهنای باند در تأخیر انتشار شبکه بیشتر باشد. پخش‌کننده حتی بعد از توقف ارسال سرویس‌دهنده به خالی کردن بافر ادامه می‌دهد، و وقتی مقدار بافر به علامت حد پائین رسید، به آن اطلاع می‌دهد که ارسال داده را از سر گیرد. محل علامت حد پائین بافر باید بگونه‌ای انتخاب شود که بافر تا رسیدن داده‌های جدید خالی نشود (buffer underrun) .

برای آن که برنامهٔ پخش بتواند سرویس‌دهندهٔ رسانه را کنترل کند، به پروتکلی مانند RTSP نیاز دارد. این پروتکل و مکانیزمهای کنترلی آن در RFC 2326 تعریف شده است (البته این پروتکل با RTP تفاوت دارد). در شکل ۷-۶۲ فرمانهای اصلی RTSP را ملاحظه می‌کنید.

فرمان	عملکرد سرویس‌دهنده
DESCRIBE	پارامترها را لیست می کند
SETUP	یک کانال بین پخش کننده و سرویس دهنده برقرار می کند
PLAY	شروع به ارسال داده می کند
RECORD	شروع به دریافت داده می کند
PAUSE	ارسال را موقتا قطع می کند
TEARDOWN	کانال را رها می کند

شکل ۷-۶۲. فرمانهای RTSP از پخش‌کننده به سرویس‌دهنده.

۷-۴-۴ رادیوی اینترنتی

همین که امکان اسال صدای جویباری (پیوسته) روی اینترنت فراهم شد، ایستگاههای رادیویی به این فکر افتادند که علاوه بر روش معمول از طریق اینترنت هم برنامه پخش کنند. کمی بعد اولین ایستگاههای رادیویی دانشگاهی روی اینترنت شروع بکار کردند؛ و بعد از آن هم ایستگاههای دانشجویی راه افتادند. با تکنولوژی امروزی، تقریباً هر کسی می‌تواند برای خود یک ایستگاه رادیویی داشته باشد. ایستگاههای رادیویی اینترنتی موضوع بسیار نو و پویایی هستند، ولی ارزش آنرا دارد که کمی دربارهٔ آن حرف بزنیم.

دو رهیافت کلی برای رادیوی‌اینترنتی (Internet radio) وجود دارد. در رهیافت اول برنامه‌ها روی دیسک ضبط می‌شوند، و شنوندگان می‌توانند برنامه‌های دلخواه را از آرشیو بار کرده و به آن گوش کنند. در حقیقت این فقط شکل دیگری از صدای جویباری است، که در قسمت قبل توضیح دادیم. همچنین می‌توان برنامه‌های زندهٔ رادیویی را روی دیسک ضبط کرد، بگونه‌ای که آرشیو فقط نیم ساعت از برنامهٔ زنده عقب باشد. مزیت این روش سادگی آن است، و می‌توان از تکنیکهایی که قبلاً تشریح کردیم برای پیاده‌سازی آن استفاده کرد.

رهیافت دیگر پخش برنامهٔ زنده روی اینترنت است. برخی از ایستگاههای رادیویی همزمان با پخش معمولی، برنامه‌های خود را روی اینترنت نیز پخش می‌کنند، ولی تعداد ایستگاههایی که فقط پخش اینترنتی دارند روز به روز در حال افزایش است. برخی از تکنیکهای صدای جویباری در اینجا نیز قابل استفاده است، ولی چند تفاوت کلیدی هم وجود دارد.

یکی از نکاتی که در هر دو سیستم یکسان است، لزوم بافر کردن داده‌ها برای حذف لرزش است. با بافر کردن ۱۰ الی ۱۵ ثانیه صدا می‌توان تا حد زیادی با قطع و وصلهای اجتناب‌ناپذیر شبکه مقابله کرد. مادامیکه بسته‌ها به موقع برسند، زمان رسیدن آنها چندان اهمیتی ندارد.

یکی از تفاوتهای مهم صدای جویباری و رادیوی اینترنتی این است که سرویس‌دهندهٔ رسانه می‌تواند داده‌ها را سریعتر از آنچه پخش‌کننده پخش می‌کند، بفرستد - چون پخش‌کننده می‌تواند با رسیدن بافر به علامت حد بالا

سرویس‌دهنده را وادار به توقف کند (یا از آن بخواهد در این مدت بسته‌های گم شده را دوباره ارسال کند). اما رادیوی اینترنتی نمی‌تواند داده‌ها را سریعتر از آنچه در حال تولید و پخش است، بفرستد.

تفاوت دیگر این است که یک رادیوی پخش زندهٔ اینترنتی معمولاً هزاران شنوندهٔ همزمان دارد، در حالیکه صدای جویباری اساساً سیستمی نقطه-به-نقطه است. در این حالت، رادیوی اینترنتی باید از پروتکلهای چندپخشی RTP/RTSP استفاده کند. این مؤثرترین روش پخش اینترنتی است.

اما در عمل رادیوهای اینترنتی به این روش کار نمی‌کنند. آنچه که معمولاً اتفاق می‌افتد این است که کاربر یک اتصال TCP به ایستگاه رادیویی برقرار کرده، و محتویات را روی این اتصال دریافت می‌کند. البته این روش مشکلاتی هم دارد، مثلاً وقتی پنجره پُر می‌شود یا بسته‌ها گم می‌شوند، پخش دچار وقفه خواهد شد.

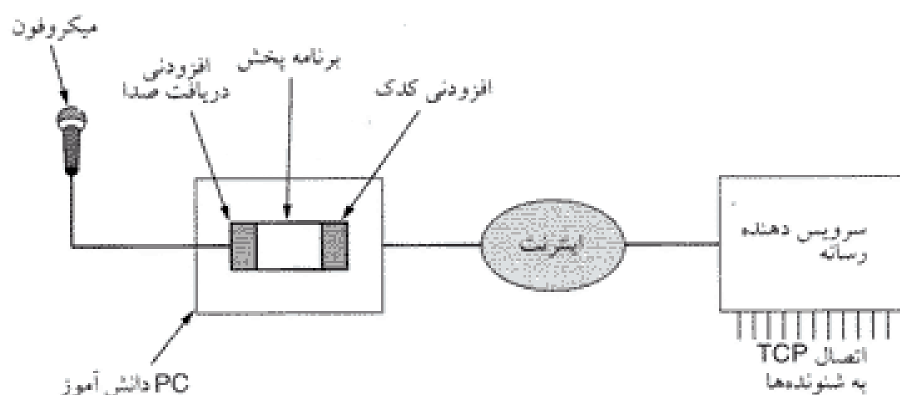
سه دلیل عمده برای استفاده از پروتکل تک‌پخشی TCP بجای پروتکل چندپخشی RTP وجود دارد. اول اینکه، اغلب ISP ها از چندپخشی پشتیبانی نمی‌کنند، و در واقع این گزینه عملی نیست. دوم اینکه، پروتکل RTP کمتر از TCP شناخته شده است، و کادر فنی رادیوهای اینترنتی (که معمولاً شرکتهای کوچکی بیش نیستند) مایلند با همان TCP معروف کار کنند. و سوم اینکه، معمولاً افراد بیشتر در محل کار به رادیوهای اینترنتی گوش می‌کنند، و این محل‌ها هم اغلب پشت دیوارآتش (firewall) قرار دارند. این دیوارهای آتش معمولاً فقط به پروتکلهای SMTP (پورت 25 - برای ایمیل)، UDP (پورت 53 - برای DNS)، و HTTP (پورت 80 - برای وب) اجازهٔ عبور می‌دهند، و جلوی هر چیز دیگر (از جمله RTP) را می‌گیرند. بنابراین، برای رادیوهای اینترنتی بهترین روش آنست که خود را یک سرویس‌دهندهٔ HTTP (که با اتصال TCP کار می‌کند) جا بزنند. البته با این کار برنامه‌های چندرسانه‌ای کارایی خود را بشدت از دست می‌دهند.

از آنجائیکه رادیوی اینترنتی موجودی نو پاست، جنگ فرمتها در آن با شدت تمام ادامه دارد. فرمتهای RealAudio ، Windows Media Audio ، و MP3 بشدت در تلاشند تا خود را بعنوان فرمت غالب رادیوی اینترنتی معرفی کنند. اخیراً فرمت دیگری بنام Vorbis نیز وارد این معرکه شده، که از نظر فنی شبیه MP3 است، ولی منبع‌باز (open source) است و آنقدر با MP3 فرق دارد که نیازی به رعایت حق اختراع آن نداشته باشد.

رادیوهای اینترنتی معمولاً یک صفحهٔ وب دارند که فهرست برنامه‌های آن (و اطلاعات متفرقهٔ دیگر، باضافهٔ مقدار زیادی آگهی) را نمایش می‌دهد. اغلب این رادیوها در حاضر از فرمتهای مختلفی پشتیبانی می‌کنند، که کاربر می‌تواند بدلخواه خود یکی از آنها را انتخاب کند. برای هر یک از این فرمتها یک آیکون جداگانه وجود دارد، که به متافایل مربوطه مرتبط است.

وقتی کاربر روی یکی از آیکونها کلیک کند، متافایل مربوطه را دریافت می‌کند. مرورگر با استفاده از اطلاعات این متافایل می‌تواند برنامهٔ کمکی مناسب را انتخاب کند. سپس این فایل را روی دیسک نوشته، و نام فایل را به برنامهٔ کمکی می‌دهد. برنامهٔ پخش این متافایل را خوانده، و URL مقصد را (که معمولاً پروتکل آن *http* است، تا بتواند دیوار آتش را دور بزند) از آن استخراج می‌کند؛ سپس به سرویس‌دهنده وصل شده، و مانند یک رادیو شروع به کار می‌کند. پروتکل *http* برای صدا (که فقط یک استریم است) کاملاً کفایت می‌کند، ولی اگر پای ویدئو (که حداقل دو استریم مستقل دارد) در میان باشد، دیگر کاری از *http* ساخته نیست و حتماً باید از چیزی شبیه *rtsp* استفاده کرد.

جذابترین جنبهٔ رادیوی اینترنتی سادگی آن است، بطوری که حتی یک دانش‌آموز دبیرستانی هم می‌تواند رادیوی اینترنتی راه بیندازد. در شکل ۷-۶۳ ساده‌ترین پیکربندی یک رادیوی اینترنتی را می‌بینید. همانطور که می‌بینید، یک رادیوی اینترنتی در واقع چیزی نیست جز یک PC معمولی با کارت صوتی و میکروفون. نرم‌افزار هم فقط یک برنامهٔ پخش رسانه (مانند Winamp) است، باضافهٔ افزودنی‌های لازم برای گرفتن صدای میکروفون و تبدیل آن به فرمت مناسب (مانند MP3 یا Vorbis).



شکل ۷-۶۳. یک رادیوی اینترنتی دانش‌آموزی.

استریم صدای خروجی، سپس، به یک سرویس‌دهندهٔ HTTP روی اینترنت فرستاده می‌شود تا در اختیار شنوندگان قرار گیرد. این قبیل سرویس‌دهنده‌ها معمولاً تعداد زیادی ایستگاه رادیویی را میزبانی می‌کنند، و حتی صفحه‌ای دارند که نشان می‌دهد در لحظهٔ حاضر کدام ایستگاهها در حال پخش برنامه‌اند. شنوندهٔ علاقمند به این سرویس‌دهنده وصل شده، ایستگاه موردنظر را انتخاب کرده و محتویات ایستگاه را از طریق اتصال TCP دریافت می‌کند. بسته‌های نرم‌افزاری مختلفی برای مدیریت یک ایستگاه رادیوی اینترنتی (از ابتدا تا انتها) وجود دارد، که از میان آنها می‌توان به icecast (که نرم‌افزاری با استاندارد منبع باز است) اشاره کرد. سرویس‌دهنده‌های زیادی هم هستند، که در ازای دریافت مبالغ ناچیزی رادیوی شما را میزبانی می‌کنند.

۷-۴-۵ صدا روی IP

آن قدیمها شبکهٔ تلفن عمومی بیشتر برای صدا بکار می‌رفت، و کمی هم داده روی آن منتقل می‌شد. اما ترافیک داده روز به روز افزایش یافت، تا اینکه در سال ۱۹۹۹ ترافیک داده و صدا مساوی شد (از آنجائیکه صدا روی ترانک‌های شبکه بصورت دیجیتال منتقل می‌شود، می‌توان آنرا هم با بیت/ثانیه اندازه گرفت). در سال ۲۰۰۲ ترافیک داده بسیار بیشتر از ترافیک صدا بود، و رشد نمایی آن همچنان ادامه دارد (در حالیکه ترافیک صدا رشد ثابتی معادل ۵٪ در سال دارد).

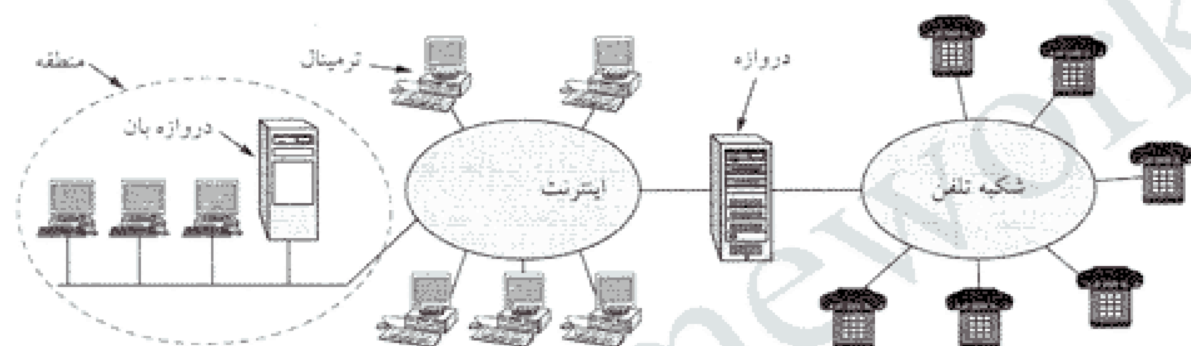
یکی از نتایج این وضعیت آن بود که بسیاری از اپراتورهای شبکه‌های سوئیچینگ بسته‌ای علاقمند شدند تا صدا را هم روی شبکه‌های داده منتقل کنند. فشاری که این کار به شبکه‌های داده می‌آورد چندان مهم نیست، چون این شبکه‌ها برای انتقال حجم عظیمی از اطلاعات طراحی شده‌اند. صورتحساب تلفن اغلب افراد بسیار بیشتر از صورتحساب اینترنت آنهاست، بهمین دلیل اپراتورهای شبکهٔ داده **تلفن اینترنتی** (Internet telephony) را یکی از راههای کسب سود سرشار ارزیابی کردند (بدون اینکه نیاز به سرمایه‌گذاری زیادی داشته باشد). بدین ترتیب بود که تلفن اینترنتی (یا همان **صدا روی IP** – voice over IP) متولد شد.

H.323

یک چیز از همان اول برای همه روشن بود: اگر هر کسی ساز خودش را بزند، این سیستم هرگز کار نخواهد کرد. برای اجتناب از چنین وضعیتی، تعدادی از شرکتهای علاقمند تحت نظارت ITU گرد آمدند، تا استانداردی برای تلفن اینترنتی وضع کنند. در سال ۱۹۹۶، ITU توصیه‌نامهٔ H.323 را با عنوان «سیستمها و تجهیزات تلفن بصری برای شبکه‌های محلی بدون تضمین کیفیت سرویس» ارائه کرد (این نامگذاری‌ها فقط از شرکتهای تلفن برمی‌آید!).

این استاندارد در سال ۱۹۹۸ مورد بازنگری قرار گرفت، و همین ویرایش H.323 بود که اولین سیستم تلفن اینترنتی بر مبنای آن ساخته شد.

H.323 بیش از آن که یک پروتکل خاص باشد، تعریفی است از معماری تلفن اینترنتی. این استاندارد بجای آن که خودش چیزی را تعریف کند، به تعدادی استانداردهای جانبی برای کُد کردن صدا، برقراری تماس، سیگنالینگ، انتقال داده، و زمینه‌های دیگر ارجاع کرده است. مدل کلی استاندارد H.323 را در شکل ۷-۶۴ ملاحظه می‌کنید. در قلب این مدل یک دروازه (gateway) قرار دارد، که اینترنت را به شبکهٔ تلفن وصل می‌کند. این مدل در سمت اینترنت به زبان پروتکلهای H.323 صحبت می‌کند، و در سمت شبکهٔ تلفن به زبان پروتکلهای PSTN . در این مدل به دستگاههای مخابراتی ترمینال (terminal) گفته می‌شود. هر شبکهٔ محلی می‌تواند دارای یک **دروازه‌بان** (gatekeeper) باشد، که ارتباطات آن **منطقه** (zone) را کنترل می‌کند.



شکل ۷-۶۴. مدل H.323 برای تلفن اینترنتی.

یک شبکهٔ تلفن به تعدادی پروتکل نیاز دارد، که اولین آنها پروتکلی برای کُد کردن صدا است. سیستم PCM ، که در فصل ۲ بررسی کردیم، در توصیه‌نامهٔ ITU G.711 تعریف شده است. این سیستم هر کانال صوتی را با نرخ نمونه‌برداری 8000 samples/sec (با نمونه‌های ۸ بیتی) کُد کرده و یک خروجی فشرده‌نشدهٔ 64 kbps بدست می‌دهد. تمام سیستمهای H.323 باید از G.711 پشتیبانی کنند، ولی پروتکلهای فشرده‌سازی صدای دیگر را هم می‌توان بکار برد (اگر چه الزامی نیست). این پروتکلها از الگوریتمهای مختلف فشرده‌سازی استفاده می‌کنند، و نسبتهای متغیری از «کیفیت/پهنای باند» بدست می‌دهند. برای مثال، G.723.1 یک بلوک ۲۴۰ نمونه‌ای (۲۴۰ بایتی – که معادل 30 msec صداست) را گرفته و با استفاده از **کُدکردن پیشگویانه** (predictive coding) به ۲۴ یا ۲۰ بایت تقلیل می‌دهد. نرخ خروجی این الگوریتم 6.4 kbps یا 5.3 kbps (ضریب کاهش ۱/۱۰ یا ۱/۱۲) می‌باشد، ضمن اینکه کیفیت صدا را هم تا حد بسیار خوبی حفظ می‌کند. کُدک‌های دیگری نیز برای کُد کردن صدا وجود دارند.

از آنجائیکه الگوریتمهای فشرده‌سازی مختلفی وجود دارند، ترمینالها باید بتوانند بطریقی پروتکل فشرده‌سازی بکار رفته را بین خود مشخص کنند. این کار بر عهدهٔ پروتکل H.245 گذاشته شده است. این پروتکل همچنین نرخ بیت اتصال را تعیین می‌کند. برای کنترل کانالهای RTP هم به پروتکل RTCP نیاز داریم. همچنین پروتکلی می‌خواهیم که کارهایی از قبیل برقراری یا قطع اتصال، ارسال بوق آزاد تلفن، تأمین زنگهای تلفن (در حالتهای مختلف)، و مانند اینها را انجام دهد. برای این کار از پروتکل ITU Q.931 استفاده می‌کنیم. ترمینالها برای ارتباط با دروازه‌بان (اگر وجود داشته باشد) هم به یک پروتکل نیاز دارند، که H.225 را برای این منظور بکار می‌بریم. این پروتکل کانال بین PC و دروازه‌بان را، که کانال RAS (Registration/Admission/Status) نام دارد، کنترل می‌کند. این کانال اجازه می‌دهد تا ترمینال به منطقه وارد شده یا آنرا ترک کند، پهنای باند را درخواست

کرده یا پس بدهد، وضعیت خود را به روز در آورد، و بسیاری کارهای دیگر. بالاخره، پروتکلی می‌خواهیم برای انتقال داده‌ها، و برای این منظور از RTP (که مدیریت آن بر عهدهٔ RTCP است) استفاده می‌کنیم. موقعیت پروتکلهای مختلف H.323 را در شکل ۷-۶۵ ملاحظه می‌کنید.

صدا	کنترل			
G.7xx / RTP	RTCP	H.225.0 (RAS)	Q.931 (سیگنالینگ تماس)	H.245 (کنترل تماس)
UDP			TCP	
IP				
پروتکل پیوند داده				
پروتکل لایه فیزیکی				

شکل ۷-۶۵. پشته پروتکل H.323 .

برای اینکه ببینید همهٔ این پروتکلها چگونه با هم کار می‌کنند، ترمینالی (که یک PC در شبکه‌ای با دروازه‌بان است) را در نظر بگیرید که می‌خواهد با تلفن راه دور تماس بگیرد. این PC ابتدا باید دروازه‌بان LAN را پیدا کند، بنابراین یک بستهٔ UDP «کشف دروازه‌بان» روی پورت 1718 در تمام شبکه پخش می‌کند. وقتی دروازه‌بان پاسخ داد، PC آدرس IP آنرا یاد می‌گیرد. اکنون PC باید خود را به دروازه‌بان بشناساند، که برای این منظور یک پیام RAS (در یک بستهٔ UDP) به آن می‌فرستد (مرحلهٔ ثبت نام – Registration). بعد از آنکه دروازه‌بان شبکه PC را قبول کرد، PC یک پیام پذیرش RAS فرستاده (مرحلهٔ پذیرش – Admission)، و تقاضای پهنای باند می‌کند. فقط بعد از اختصاص پهنای باند از سوی دروازه‌بان است، که تماس تلفنی می‌تواند شروع شود. این مرحله برای تضمین حداقل کیفیت سرویس صورت می‌گیرد، تا دروازه‌بان بتواند تعداد تماسهای همزمان را کنترل کرده و از فشار بیش از حد به خطوط خروجی جلوگیری کند.

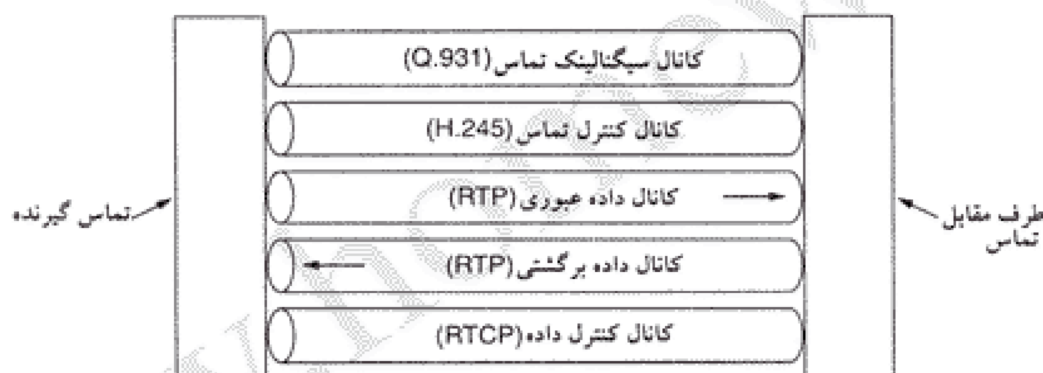
پس از آن، PC برای شروع تماس تلفنی یک اتصال TCP به دروازه‌بان برقرار می‌کند. از آنجائیکه برای برقراری تماس به پروتکلهای شبکهٔ تلفن نیاز است و این پروتکلها هم اتصال-گرا هستند، فقط از TCP می‌توان برای این منظور استفاده کرد. از طرف دیگر، چون در شبکهٔ تلفن چیزی شبیه RAS وجود ندارد، در انتخاب نوع پروتکل آن قسمت (UDP یا TCP) آزادی عمل داریم؛ و بعلت کمتر بودن سربارهٔ UDP ، طراحان H.323 آنرا انتخاب کرده‌اند.

پس از گرفتن پهنای باند لازم، PC می‌تواند یک پیام Q.931 *SETUP* روی اتصال TCP بفرستد، و شماره تلفن مقصد (یا آدرس IP و شمارهٔ پورت، اگر مقصد یک کامپیوتر باشد) را به دروازه‌بان بدهد. دروازه‌بان هم برای تصدیق دریافت درخواست کاربر، یک پیام Q.931 *CALL PROCEEDING* به آن برمی‌گرداند. سپس، دروازه‌بان پیام *SETUP* را به دروازه هدایت می‌کند.

دروازه، که نیمی کامپیوتر و نیمی سوئیچ تلفن است، شمارهٔ تلفن خواسته شده را می‌گیرد. ایستگاه پایانی که تلفن در آن قرار دارد، به شمارهٔ مزبور زنگ می‌زند و یک پیام Q.931 *ALERT* هم به PC مبدأ می‌فرستد تا نشان دهد که زنگ خوردن شروع شده است. وقتی طرف مقابل گوشی را برمی‌دارد، ایستگاه پایانی یک پیام Q.931 *CONNECT* به مبدأ برمی‌گرداند تا بگوید که ارتباط برقرار شده است.

بعد از برقراری تماس دروازه‌بان دیگر کاری ندارد و از مدار خارج می‌شود، ولی دروازه همچنان به کار خود ادامه خواهد داد. بسته‌های بعدی دروازه‌بان را دور زده، و مستقیماً به آدرس IP دروازه فرستاده می‌شوند. در این نقطه دیگر فقط یک کانال ارتباطی معمولی بین مبدأ و مقصد وجود دارد، که چیزی نیست جز اتصال لایه‌های فیزیکی برای انتقال بیت‌ها، بدون هیچ اطلاعی از ماهیت واقعی کار.

در این مرحله برای تعیین پارامترهای تماس تلفنی از پروتکل H.245 استفاده می‌شود. این پروتکل از کانالهای H.245 استفاده می‌کند، که همیشه باز هستند. هر دو طرف ابتدا روی این کانال مقدورات خود را اعلام می‌کند، برای مثال نوع تماس (مثلاً، ویدئو - چون H.323 قادر به انتقال ویدئو نیز هست - یا تماس کنفرانسی)، نوع کُدک، و غیره. همین که دو طرف از توانائیهای یکدیگر با خبر شدند، دو کانال یک طرفهٔ داده (با تعیین نوع کُدک و سایر پارامترها) بین آنها ایجاد می‌شود. از آنجائیکه امکان دارد قابلیتهای دو طرف یکسان نباشد، کانالهای رفت و برگشت می‌توانند کاملاً متفاوت باشند. بعد از پایان مرحلهٔ مذاکره بر سر پارامترها، انتقال داده با استفاده از RTP می‌تواند شروع شود، که برای مدیریت آن از RTCP کمک می‌گیریم (این پروتکل نقش مهمی در کنترل ازدحام دارد). اگر ویدئو نیز وجود داشته باشد، سنکرون کردن صدا و تصویر هم بر عهدهٔ RTCP خواهد بود. انواع کانالهای ممکن را در شکل ۷-۶۶ ملاحظه می‌کنید. در پایان تماس نیز، برای قطع ارتباط از سیگنالهای Q.931 استفاده می‌شود.



شکل ۷-۶۶. کانال‌های منطقی بین دو طرف تماس.

بعد از قطع ارتباط، PC تماس‌گیرنده یک پیام RAS دیگر به دروازه‌بان می‌فرستد تا پهنای باند اختصاص داده شده را به آن پس بدهد (یا تماس دیگری بگیرد).

دربارهٔ **کیفیت سرویس** (Quality of Service - QoS)، که نقش اساسی در موفقیت VOIP دارد، هیچ حرفی نزدیم. علت آن است که QoS جزء وظایف H.323 نیست. اگر شبکهٔ موجود بتواند (با استفاده از تکنیکهای فصل ۵) ارتباطی پایدار و بدون لرزش بین PC تماس‌گیرنده و دروازه برقرار کند، QoS تماس خوب خواهد بود؛ در غیر اینصورت، خیر. شبکهٔ تلفن از PCM استفاده می‌کند، و هیچوقت لرزش ندارد.

SIP - پروتکل آغاز نشست

H.323 توسط ITU طراحی شد، و بسیاری از اینترنتی‌ها آنرا یک محصول تلفنی دیگر (بزرگ، پیچیده، و انعطاف‌ناپذیر) یافتند. بهمین دلیل، IETF کمیته‌ای تشکیل داد تا روشی ساده‌تر و مدولارتر برای VOIP طراحی کند. ماحصل کار این کمیته SIP (پروتکل آغازنشست - Session Initiation Protocol) بود که در RFC 3261 تعریف شده است. در این پروتکل نحوهٔ برقراری تماسهای تلفن اینترنتی، کنفرانس ویدئویی، و ارتباطات چندرسانه‌ای دیگر تشریح شده است. برخلاف H.323 که مجموعه‌ایست از چند پروتکل، SIP یک ماژول منفرد

است، ولی بگونه‌ای طراحی شده که بتواند با برنامه‌های اینترنتی موجود کار کند. برای مثال، در این پروتکل شماره‌های تلفن بصورت URL تعریف شده‌اند، و می‌توان آنها را روی صفحه‌های وب قرار داد؛ با کلیک کردن این لینک‌ها فرآیند برقراری تماس شروع می‌شود (درست به همان شکلی که لینکهای *mailto* کار می‌کنند).

پروتکل SIP می‌تواند نشست‌های دو-طرفه (تماسهای تلفنی معمولی)، نشست‌های چند-طرفه (کنفرانس تلفنی - که تمام شرکت‌کنندگان می‌توانند حرف بزنند و حرف دیگران را بشنوند)، و نشست‌های چندپخشی (یک گوینده و چندین شنونده) برقرار کند. در این نشست‌ها می‌توان صدا، تصویر و داده منتقل کرد (که این آخری به درد بازیهای گروهی روی اینترنت می‌خورد). البته SIP فقط شروع، کنترل و قطع نشست‌ها را بر عهده دارد، و انتقال داده را پروتکلهای دیگر، مانند RTP/RTCP ، انجام می‌دهند. از آنجائیکه SIP یک پروتکل لایهٔ کاربرد است، می‌تواند روی TCP یا UDP نیز اجرا شود.

پروتکل SIP سرویسهای متنوعی ارائه می‌کند، از جمله یافتن تماس‌شونده (اگر در خانه نباشد)، تعیین قابلیتهای تماس‌شونده، و مکانیزمهایی برای شروع و قطع ارتباط. در ساده‌ترین شکل، SIP یک نشست بین کامپیوتر تماس‌گیرنده و کامپیوتر تماس‌شونده برقرار می‌کند، و ما هم ابتدا همین فرآیند را بررسی خواهیم کرد.

در SIP شماره تلفن‌ها با یک URL (که از پروتکل *sip:* استفاده می‌کند) مشخص می‌شوند؛ مثلاً، *sip:ilse@cs.university.edu* شماره تلفن کاربری بنام *ilse* در ناحیهٔ *cs.university.edu* است. در URL های SIP می‌توان از آدرسهای IPv4 ، آدرسهای IPv6 ، یا شماره تلفن واقعی استفاده کرد.

SIP یک پروتکل متنی (بر اساس مدل HTTP) است، که در آن نام متد در خط اول می‌آید، و بدنبال آن پارامترهای مورد نیاز فرستاده می‌شوند. بسیاری از سرآیندهای SIP از MIME گرفته شده‌اند، تا این پروتکل بتواند با برنامه‌های اینترنتی موجود کار کند. در شکل ۷-۶۷ شش تا از متدهای SIP ، که در مشخصهٔ اصلی آن تعریف شده‌اند، را ملاحظه می‌کنید.

متد	توضیح
INVITE	درخواست برقراری یک نشست
ACK	تایید شروع نشست
BYE	درخواست پایان نشست
OPTIONS	پرس و جو درباره قابلیتهای میزبان
CANCEL	لغو درخواست معلق مانده
REGISTER	دادن اطلاعات مکان مشتری به سرویس دهنده

شکل ۷-۶۷. متدهای اصلی SIP .

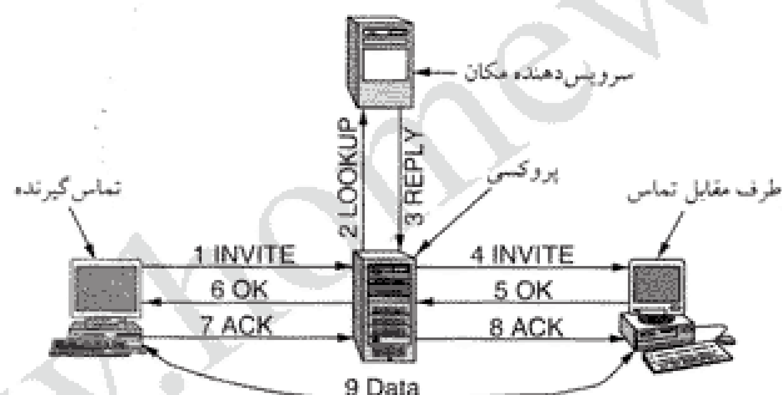
برای برقراری یک نشست دو روش وجود دارد: برقرای اتصال TCP به تماس‌شونده و ارسال یک پیام *INVITE* به آن؛ ارسال پیام *INVITE* در یک بستهٔ UDP . در هر دو حالت، سرآیند خط دوم و خطهای بعدی ساختار بدنهٔ پیام (شامل قابلیتهای تماس‌گیرنده، نوع رسانه، و فرمتهای آن) را مشخص می‌کنند. اگر تماس‌شونده دعوت به تماس را قبول کند، یک کُد پاسخ شبیه HTTP (کُدهای سه رقمی، که در اینجا هم 200 نشانهٔ قبول درخواست است) برمی‌گرداند. پس از این کُد پاسخ، تماس‌شونده می‌تواند اطلاعات مربوط به خود (قابلیتها، نوع رسانه، و فرمتها) را بفرستد.

برای برقراری هر تماس سه پیام باید رد و بدل شود، بنابراین تماس‌گیرنده یک پیام *ACK* به تماس‌شونده برمی‌گرداند تا نشان دهد که پذیرش وی را دریافت کرده است؛ این پایان پروتکل برقراری نشست است.

هر یک از طرفین تماس می‌توانند برای قطع ارتباط پیام *BYE* بفرستند. وقتی طرف مقابل این پیام را تصدیق کرد، نشست خاتمه خواهد یافت.

هر ماشین می‌تواند برای تعیین قابلیتهای خود (و اینکه اصلاً قادر به برقرای تماس VOIP هست یا نه) از متد *OPTIONS* استفاده کند، و معمولاً این کار را قبل از شروع نشست انجام می‌دهد.

پروتکل SIP از متد *REGISTER* برای یافتن افرادی که در محل مورد انتظار نیستند، استفاده می‌کند. هر ماشین پیام *REGISTER* خود را به یک **سرویس‌دهندهٔ مکان** SIP (SIP location server)، که محل افراد را تعقیب می‌کند، می‌فرستد. وقتی کسی می‌خواهد با شما تماس بگیرد، می‌تواند به کمک این سرویس‌دهنده محل فعلی شما را پیدا کند. روش کار را در شکل ۷-۶۸ می‌بینید. در اینجا، تماس‌گیرنده پیام *INVITE* خود را به یک پروکسی فرستاده است (هدف از بکارگیری پروکسی برداشتن وظیفهٔ یافتن مکان مخاطب از دوش کامپیوتر تماس‌گیرنده است). این پروکسی به کمک سرویس‌دهندهٔ مکان محل فرد مورد نظر را یافته، و پیام *INVITE* را به آن می‌فرستد (تمام پیامهای بعدی هم با واسطهٔ همین پروکسی رد و بدل خواهند شد). پیامهای *LOOKUP* و *REPLY* جزء SIP نیستند، و از هر پروتکلی (که با سرویس‌دهندهٔ مکان مورد استفاده سازگار باشد) می‌توان برای این منظور استفاده کرد.



شکل ۷-۶۸. استفاده از سرویس‌دهندهٔ پروکسی با SIP .

پروتکل SIP ویژگیهای دیگری (از جمله انتظار مکالمه، پیگیری تماس، رمزنگاری، و احراز هویت) نیز دارد که در اینجا به آنها نخواهیم پرداخت. اگر دروازهٔ مناسب بین اینترنت و شبکهٔ تلفن وجود داشته باشد، SIP می‌تواند بین کامپیوتر و تلفنهای معمولی نیز تماس برقرار کند.

مقایسهٔ H.323 و SIP

H.323 و SIP شباهتها و تفاوتهای بسیاری دارند. هر دوی آنها می‌توانند تماسهای مستقیم یا کنفرانسی بین کامپیوترها و تلفنهای معمولی برقرار کنند. هر دو از مذاکره برای تعیین پارامترهای تماس، رمزنگاری، و پروتکلهای RTP/RTCP پشتیبانی می‌کنند. فهرستی از شباهتها و تفاوتهای این دو را در شکل ۷-۶۹ ملاحظه می‌کنید.

با وجود شباهتهای ظاهری، این دو پروتکل از نظر فلسفهٔ وجودی بسیار با هم متفاوتند. H.323 پروتکلی است بزرگ، پیچیده و استاندارد صنعت تلفن، که دقیقاً مشخص کرده چه چیزهایی مجازند و چه چیزهایی ممنوع. در این رهیافت همه چیز کاملاً مشخص و تعریف شده، و ارتباط بین سیستمهای مختلف بسادگی امکانپذیر است. بهایی که برای این سادگی باید پرداخت، عبارتست از بزرگی، پیچیدگی و انعطاف‌ناپذیری، که انطباق این پروتکل با نیازهای آینده را دشوار کرده است.

آیتم	H.323	SIP
مسئول طراحی	ITU	IETF
سازگاری با PSTN	بلی	تا حد زیاد
سازگاری با اینترنت	خیر	بلی
معماری	یکپارچه	ماژولار
کامل بودن	پشته پروتکل کامل	فقط برقراری تماس
مذاکره پارامترها	بلی	بلی
سیگنالینگ تماس	Q.931 رویTCP	SIP روی TCP یا UDP
فرمت پیام	باینری	متنی
انتقال رسانه	RTP/RTCP	RTP/RTCP
تماس چند طرفه	بلی	بلی
کنفرانس چند رسانه ای	بلی	بلی
آدرس دهی	شماره تلفن یا میزبان	URL
پایان تماس	صریح یا رها کردنTCP	صریح یا بعد از انقضای زمان
پیام رسانی فوری	خیر	بلی
رمزنگاری	بلی	بلی
اندازه استاندارد	۱۴۰۰ صفحه	۲۵۰ صفحه
پیاده سازی	بزرگ و پیچیده	متوسط
وضعیت فعلی	کاربرد گسترده	درحال رشد

شکل ۷-۶۹. مقایسه‌ای بین H.323 و SIP .

از طرف دیگر، SIP یک پروتکل سبک و معمولی اینترنتی است که با مبادلهٔ پیامهای متنی کار می‌کند، و سازگاری خوبی با پروتکلهای موجود اینترنت دارد، ولی در زمینهٔ ارتباط با پروتکلهای سیگنالینگ تلفن چندان قوی نیست. از آنجائیکه IETF مدل VOIP خود را بصورت کاملاً مدولار تعریف کرده، این پروتکل انعطاف‌پذیری بالایی دارد و براحتی می‌توان آنرا با نیازهای آتی انطباق داد. نقطه ضعف این رهیافت مشکلات ناشی از ناسازگاری سیستمهاست، که IETF سعی کرده با برپایی سمینارهای متعدد و تبادل آراء بین سازندگان مختلف آنرا به حداقل برساند.

صدا روی IP (VOIP) مبحثی نو و در حال تحول است. کتابهای متعددی در این زمینه نوشته شده، که از میان آنها می‌توان به (Colins, 2001; Davidson and Peters, 2000; Kumar et al., 2001; and Wright, 2001) اشاره کرد. در شمارهٔ May/June 2002 مجلهٔ *Internet Computing* نیز چندین مقاله در زمینهٔ VOIP ارائه شده است.

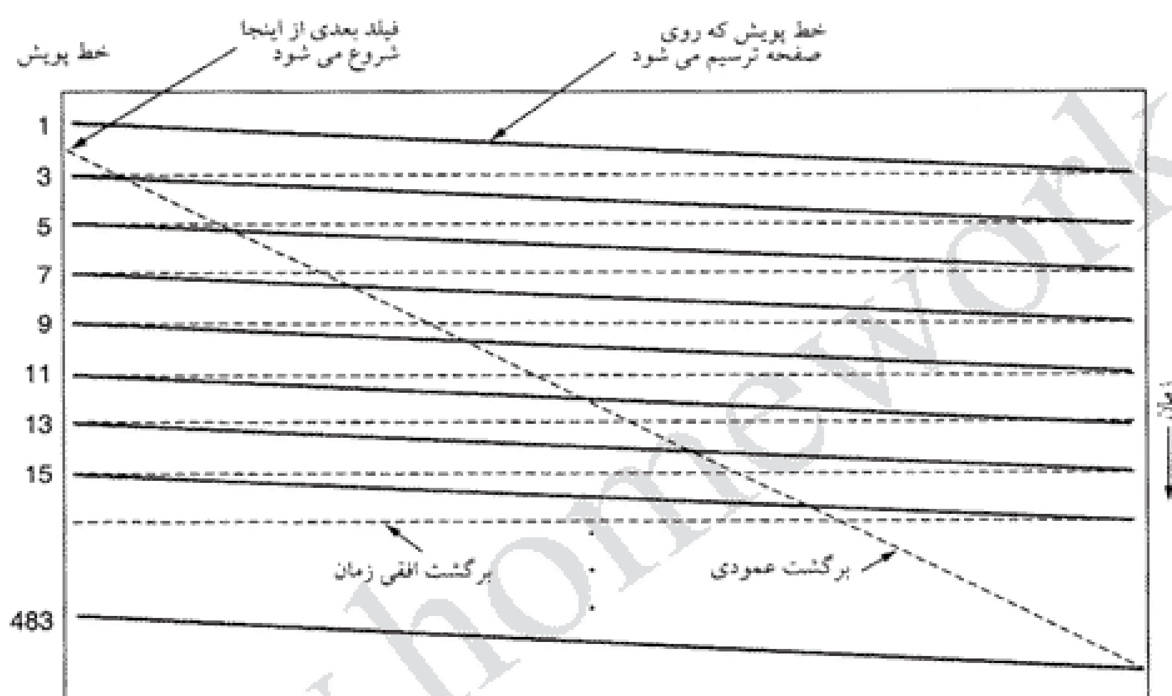
۷-۴-۶ مقدمه‌ای بر ویدئو

تا اینجا دربارهٔ گوش صحبت کردیم؛ اکنون وقت آن است که به چشم بپردازیم (نگران نباشید بعد از آن دیگر سراغ بینی نخواهیم رفت!). یکی از ویژگیهای چشم انسان این است که وقتی تصویری روی شبکیه می‌افتد، این تصویر برای چند هزارم ثانیه باقی می‌ماند. اگر یک تصویر را خط به خط رسم کنیم بطوریکه تمام این خطها در کمتر از یک پنجاهم ثانیه رسم شوند، چشم متوجه قطعه قطعه بودن تصویر نخواهد شد. تمام وسایل تصویری (از جمله تلویزیون) از این ویژگی برای تولید تصاویر متحرک استفاده می‌کنند.

سیستمهای آنالوگ

برای درک بهتر ویدئو، بهتر است از ساده‌ترین وسیله یعنی یک تلویزیون سیاه-سفید قدیمی کمک بگیریم.

برای تبدیل یک تصویر دو-بُعدی به سیگنالی یک-بُعدی (تابعی از ولتاژ بر حسب زمان)، دوربین تلویزیونی با شروع از بالای تصویر آن را به وسیلهٔ یک پرتو الکترونی از چپ براست اسکن کرده و بتدریج پائین می‌آید، و در هر لحظه شدت نور را ثبت می‌کند. وقتی پرتو الکترونی به انتهای اسکن (که فریم نامیده می‌شود) رسید، دوباره به نقطهٔ شروع برمی‌گردد. خروجی دوربین همین سیگنال (تابع شدت نور بر حسب زمان) است، و گیرنده با تکرار فرآیند اسکن دوباره تصویر را می‌سازد. روش اسکن کردن تصویر (که در دوربین و گیرنده یکسان است) در شکل ۷-۷۰ نشان داده شده است.



شکل ۷-۷۰. الگوی اسکن کردن تصویر در سیستم ویدئویی NTSC .

پارامترهای اسکن کردن تصویر از کشوری به کشور دیگر فرق می‌کند. در سیستمهای ویدئویی آمریکای شمالی، جنوبی و ژاپن از اسکن ۵۲۵ خطی، نسبت افقی-به-عمودی 4:3 ، و سرعت ۳۰ فریم بر ثانیه استفاده می‌شود. سیستمهای اروپایی اسکن ۶۲۵ خطی، نسبت افقی-به-عمودی 4:3 ، و سرعت ۲۵ فریم بر ثانیه را بکار می‌برند. در هر دو سیستم، برای چهارگوش کردن تصویر در لامپهای گرد قدیمی، چند خط از بالا و چند خط از پائین اصلاً نمایش داده نمی‌شود؛ در NTSC (که ۵۲۵ خط دارد) فقط ۴۸۳ خط نشان داده می‌شود، و در سیستمهای اروپایی (PAL/SECAM) ۵۷۶ خط (از ۶۲۵ خط). در مدت برگشت پرتو الکترونی به نقطهٔ شروع این پرتو خاموش است (تا روی تصویر خط نیندازد)، و بسیاری از کشورها (بویژه در اروپا) از این فرصت برای ارسال اطلاعات متنی (اخبار، وضع هوا، اخبار ورزشی، قیمت سهام، و غیره) استفاده می‌کنند؛ این سرویس به **تله‌تکست** (TeleText) معروف است.

با اینکه ۲۵ فریم بر ثانیه برای نمایش تصاویر متحرک کافیست، اما برخی افراد (بویژه سالخورده‌گان) در این سرعت احساس می‌کنند تصاویر چشمک می‌زند (چون مدت دوام تصویر روی شبکیهٔ آنها کمتر از افراد معمولی است). برای حل این مشکل، بجای بالا بردن نرخ فریم (که باعث هدر رفتن پهنای باند خواهد شد) از تکنیک متفاوتی استفاده می‌شود. در این تکنیک (بجای نمایش خطوط اسکن متوالی)، ابتدا خطوط فرد و سپس خطوط

زوج نمایش داده می‌شوند. بدین ترتیب، در هر بار حرکت پرتو الکترونی از بالا تا پائین صفحه فقط نیمی از یک فریم کامل نشان داده می‌شود؛ این نیم‌فریم را یک **فیلد** (field) می‌گویند. آزمایشات نشان داده که با وجود احساس چشمک زدن تصویر در سرعت ۲۵ فریم بر ثانیه، این احساس در سرعت ۵۰ فیلد بر ثانیه از بین می‌رود. این تکنیک **خط-در-میانی** (interlacing) نام دارد. به تلویزیونها یا ویدئوهای غیر خط-در-میانی **پیشرونده** (progressive) گفته می‌شود. توجه کنید که، با اینکه فیلمهای سینمایی با سرعت ۲۴ فریم بر ثانیه پخش می‌شوند، ولی در آنجا هر تصویر به مدت ۱/۲۴ ثانیه بطور کامل دیده می‌شود.

ویدئوی رنگی از همان الگوی اسکن تک‌رنگ (سیاه-سفید) استفاده می‌کند، ولی در آن برای هر رنگ اصلی یک پرتو الکترونی مستقل وجود دارد که بطور هماهنگ عمل می‌کنند. رنگهای اصلی عبارتند از: قرمز، سبز، آبی (RGB) . همانطور که می‌دانید، هر رنگی را می‌توان با برهم‌نهی خطی رنگهای اصلی (با شدت‌های مناسب) ایجاد کرد. با این حال، برای انتقال تصاویر رنگی روی یک کانال، باید سیگنالهای رنگ را در یک **سیگنال مرکب** (composite signal) ترکیب کرد.

وقتی تلویزیون رنگی اختراع شد، تکنیکهای متفاوتی برای نمایش رنگ وجود داشت، و هر کشور یکی از این تکنیکها را برای خود انتخاب کرد، که این منجر شد به سیستمهای تلویزیون رنگی ناسازگار (وضعیتی که همچنان ادامه دارد). (توجه داشته باشید که این قضیه هیچ ارتباطی با دعوای VHS ، بتاماکس، و P2000 - که سیستمهای ضبط تصاویر رنگی هستند - ندارد.) اما در تمام کشورها یک الزام قانونی وجود داشت: تصاویر رنگی باید روی گیرنده‌های سیاه-سفید هم قابل دریافت باشند. به همین دلیل کُد کردن جداگانهٔ سیگنالهای RGB (که ساده‌ترین روش ارسال سیگنالهای رنگی است) عملاً کنار گذاشته شد. (البته RGB کارآمدترین روش کُد کردن سیگنالهای رنگی نیست.)

اولین سیستم رنگی در آمریکا توسط «کمیتهٔ ملی استانداردهای تلویزیون» (National Television Standards Committee) استاندارد شد، و نام خود را هم به آن داد: NTSC . تلویزیون رنگی سالها بعد وارد اروپا شد، زمانی که تکنولوژی آن پیشرفت قابل توجهی کرده بود، به همین دلیل سیستمهای اروپایی از نظر رنگ و مقاومت در برابر نویز بسیار بهتر از سیستمهایی آمریکایی بودند. در اروپا دو سیستم تلویزیون رنگی بکار گرفته شد: SECAM (SEquential Couleur Avec Memoire) که در فرانسه و اروپای شرقی بکار گرفته شد، و PAL

(Phase Alternating Line) که در بقیهٔ اروپا از آن استفاده می‌شود. تفاوت کیفیت رنگ بین NTSC و PAL/SECAM چنان فاحش است، که NTSC را به استهزاء «**هر دفعه به یک رنگ**» (Never Twice the Same Color) هم می‌گویند.

برای آن که سیگنالهای RGB روی تلویزیونهای سیاه-سفید هم قابل دریافت باشند، در هر سه سیستم سیگنالهای RGB بصورت خطی - با نسبتهای مختلف - در یک سیگنال **روشنایی** (luminance) و دو سیگنال **رنگ** (chrominance) ترکیب می‌شوند. جالبست بدانید که حساسیت چشم انسان نسبت به سیگنالهای روشنایی بسیار بیشتر از سیگنالهای رنگ است، بنابراین دقت چندانی در ارسال سیگنالهای رنگ لازم نیست. سیگنال روشنایی با همان فرکانس تلویزیونهای سیاه-سفید قدیمی پخش می‌شود، تا آنها هم بتوانند تصاویر را دریافت کنند؛ سیگنالهای رنگ نیز با فرکانسهای بالاتر (در باندی باریک) پخش می‌شوند. در برخی از تلویزیونها کنترلهایی بنام روشنایی، رنگ و غلظت رنگ وجود دارد، که در واقع این سیگنالها را کنترل می‌کنند. درک روشنایی و رنگ برای فهم تکنیکهای فشرده‌سازی ویدئو ضروری است.

در سالهای اخیر سر و صدای زیاد در اطراف HDTV (تلویزیون باوضوح بالا – High Definition TeleVision) بر پا شده است. این تلویزیونها با (تقریباً) دو برابر کردن تعداد خطوط اسکن، تصاویر بسیار بهتری تولید می‌کنند. آمریکا، اروپا و ژاپن همگی سیستمهای HDTV خاص خود را توسعه داده‌اند، که هیچکدام با دیگری سازگار نیست. (انتظار دیگری داشتید؟) اصول کار HDTV (اسکن، روشنایی، رنگ، و غیره) با سیستمهای موجود یکسان است، ولی نسبت افقی–به–عمودی در همهٔ آنها 16:9 می‌باشد. این فرمت برای نمایش فیلمهای سینمایی (که روی فیلمهای 35 mm با نسبت 3:2 ضبط می‌شوند) بسار مناسبتر است.

سیستمهای دیجیتال

ساده‌ترین راه نمایش ویدئوی دیجیتال عبارتست از توالی فریمهایی که هر کدام از تعدادی پیکسل (pixel) با آرایش مستطیلی تشکیل شده‌اند. هر پیکسل را می‌توان با یک بیت (سیاه یا سفید) نشان داد. کیفیت این سیستم شبیه ارسال تصویر با فکس است (فوق‌العاده وحشتناک!).

اگر از ۸ بیت برای نمایش هر پیکسل استفاده کنیم، می‌توانیم ۲۵۶ سایهٔ خاکستری را نشان دهیم. کیفیت سیاه–سفید چنین سیستمی نسبتاً خوب است. در سیستمهای رنگی خوب، برای هر یک از رنگهای RGB از ۸ بیت جداگانه استفاده می‌شود، اگر چه تمام این سیستمها رنگها را هنگام ارسال به یک سیگنال مرکب تبدیل می‌کنند. استفاده از ۲۴ بیت برای هر پیکسل تعداد رنگهای ممکن را به حدود ۱۶ میلیون محدود می‌کند، ولی هیچ چشمی قادر به تشخیص این تعداد رنگ نیست (بیشتر که جای خود دارد). تصاویر رنگی دیجیتال با استفاده از سه پرتو اسکن‌کننده (یک پرتو برای هر رنگ) تولید می‌شوند. روش کار شبیه شکل ۷–۷۰ است، با این تفاوت که پیکسلهای منفرد جای خطوط پیوسته را گرفته‌اند.

در ویدئوی دیجیتال هم (مانند آنالوگ) برای ایجاد تصاویر متحرک باید حداقل ۲۵ فریم بر ثانیه نمایش داده شود. اما از آنجائیکه مانیتورهای امروزی قادرند تا ۷۵ تصویر در هر ثانیه نمایش دهند، نیازی به استفاده از تکنیک خط–در–میانی نیست (و معمولاً هم استفاده نمی‌شود). وقتی بتوانیم هر فریم را سه بار پشت سر هم روی مانیتور رسم کنیم، دیگر چیزی بنام **چشمک** (flicker) وجود نخواهد داشت.

به تفاوت این دو مفهوم توجه کنید: نرمی حرکت به تعداد تصاویر مختلف در هر ثانیه بستگی دارد، در حالیکه چشمک به تعداد دفعات ترسیم یک تصویر روی صفحه وابسته است. در یک تصویر ثابت که با سرعت ۲۰ فریم بر ثانیه نمایش داده می‌شود، هیچ مشکلی بنام نرمی حرکت وجود ندارد، ولی همین تصویر دارای لرزش و چشمک است، چون هر تصویر برای مدت زمان کافی روی شبکیه نقش نمی‌بندد. حال اگر فیلمی با سرعت ۲۰ فریم بر ثانیه نمایش پخش شود، ولی هر تصویر ۴ بار روی صفحهٔ مانیتور رسم شود، تصاویر پخش شده بدون چشمک خواهند بود ولی حرکت نرم نیست.

اهمیت این دو پارامتر وقتی بیشتر روشن می‌شود که پهنای باند لازم برای ارسال تصاویر ویدئویی دیجیتال روی شبکه را در نظر بگیریم. مانیتورهای امروزی همچنان دارای نسبت افقی–به–عمودی 4:3 هستند، چون باید از لامپهای تلویزیونی که تولید انبوه شده‌اند، استفاده کنند. وضوح این مانیتورها اغلب 768 × 1024، 960 × 1280 یا 1200 × 1600 است. حتی کمترین وضوح (768 × 1024) با ۲۴ بیت بر پیکسل، و سرعت ۲۵ فریم بر ثانیه به پهنای باندی معادل 472 Mbps نیاز دارد. این پهنای باند معادل کاریر SONET OC-12 است، و فعلاً که قرار نیست به هر خانه یک خط OC-12 بکشند. اگر بخواهیم برای حذف چشمک این سرعت را دو برابر کنیم، که اوضاع خیلی خرابتر خواهد شد. البته برای حذف چشمک می‌توان فریمها را در تلویزیون ذخیره کرده و هر فریم را ۲ بار روی صفحه رسم کرد. اما مشکل اینست که تلویزیونهای معمولی حافظه ندارند، و اگر هم داشتند، سیگنالهای آنالوگ را نمی‌توان بدون تبدیل به دیجیتال در حافظه ذخیره کرد (و همهٔ اینها یعنی هزینهٔ اضافی).

۷-۴-۷ فشرده‌سازی ویدئو

تا اینجا فهمیدیم که به ارسال تصاویر ویدئویی فشرده‌نشده حتی نباید فکر کرد؛ تنها امیدمان اینست که بتوانیم تصاویر ویدئویی را به میزان زیاد فشرده کنیم. خوشبختانه کار تحقیقاتی گسترده در چند دههٔ گذشته باعث شد تا ارسال تصاویر ویدئویی فشرده روی شبکه ممکن شود. در این قسمت خواهید دید که فشرده کردن تصاویر ویدئویی چگونه انجام می‌شود.

تمام سیستمهای مبتنی بر داده‌های فشرده دو بخش دارند: یک الگوریتم فشرده‌سازی در مبدأ، و یکی برای عکس آن در مقصد. در ادبیات فنی به این الگوریتمها بترتیب **کُدکردن** (encoding) و **دیکُدکردن** (decoding) می‌گویند؛ ما هم از همین اصطلاحات استفاده خواهیم کرد.

در الگوریتمهای فشرده‌سازی نوعی عدم تقارن ذاتی وجود دارد، که برای درک بهتر این فرآیند از اهمیت زیادی برخوردار است. اول اینکه، یک سند چندرسانه‌ای (مثلاً، یک فیلم سینمایی) فقط یک بار کُد شده و روی سرویس‌دهنده قرار داده می‌شود، ولی دیکُد شدن آن هزاران بار (هر بار که یکی از کاربران می‌خواهد آنرا تماشا کند) اتفاق می‌افتد. این بدان معناست که الگوریتم کُد کردن می‌تواند کُند و متکی به سخت‌افزارهای گران‌قیمت باشد، مشروط باینکه الگوریتم دیکُد کردن سریع بوده و نیازی به سخت‌افزارهای گران‌قیمت نداشته باشد. برای شرکتهای پخش چندرسانه‌ای دو هفته کرایه کردن یک اَبَرکامپیوتر (و کُد کردن تمام فیلمهایی که دارند) چندان دور از منطق نیست، ولی نمی‌توان از کاربران انتظار داشت برای تماشای یک فیلم ویدئویی دو ساعت اَبَرکامپیوتر اجاره کنند. در بسیاری از سیستمهای فشرده‌سازی الگوریتم کُد کردن، به قیمت سریع و ساده شدن عملیات دیکُد، پیچیده و وقت‌گیر طراحی شده‌اند.

از طرف دیگر، در سیستمهای **چندرسانه‌ای بی‌درنگ** (real-time multimedia) – مانند کنفرانس ویدئویی – کُند بودن فرآیند کُد کردن نیز غیرقابل‌قبول است. در اینجا کُد کردن باید در-لحظه و بصورت بی‌درنگ انجام شود. در نتیجه، چنین سیستمهایی باید از الگوریتمهای متفاوتی استفاده کنند.

عدم تقارن دیگر در فرآیند کُد/دیکُد اینست که آنها نباید الزاماً عکس یکدیگر باشند. این بدان معناست که در سیستمهای چندرسانه‌ای الزامی نیست فایلی که در مقصد دیکُد می‌شود دقیقاً همان فایلی باشد که در مبدأ کُد شده، و می‌تواند چیزهایی را از دست بدهد. وقتی خروجی دیکُد شده دقیقاً همان ورودی کُد شده نباشد، اصطلاحاً گفته می‌شود که سیستم **تلفات‌دار** (lossy) است. در مقابل، سیستمی که خروجی دقیقاً معادل ورودی باشد، **بدون تلفات** (lossless) نامیده می‌شود. سیستمهای تلفات‌دار از اهمیت زیادی برخوردارند، چون از دست دادن مقدار کمی از اطلاعات به فشرده‌سازی بسیار بالایی منجر می‌شود.

استاندارد JPEG

ویدئو عبارتست از توالی چند تصویر (بعلاوهٔ صدا). اگر بتوانیم برای کُد کردن تصاویر ثابت الگوریتم مناسبی پیدا کنیم، می‌توانیم آنرا برای تصاویر متحرک (ویدئو) نیز بکار ببریم. امروزه الگوریتمهای خوبی برای فشرده کردن تصاویر ثابت وجود دارد، پس اجازه دهید از همانها شروع کنیم. استاندارد JPEG (Joint Photographic Experts Group) توسط گروهی از متخصصان عکاسی (تحت نظارت ITU ، ISO و IEC و چند سازمان دیگر) برای فشرده کردن تصاویر غیرگرافیکی (بویژه، عکس) توسعه داده شد. اهمیت این الگوریتم در آنجاست که مهمترین استاندارد فشرده‌سازی تصاویر متحرک یعنی MPEG ، در واقع چیزی نیست جز کُد کردن فریمهای متوالی با JPEG (باضافهٔ چند ویژگی دیگر برای فشرده‌کردن بین فریمها و تشخیص حرکت). JPEG در استاندارد ISO 10918 تعریف شده است.

الگوریتم JPEG دارای چهار حالت و چندین گزینه است (و در واقع بیشتر به یک لیست خرید می‌ماند، تا

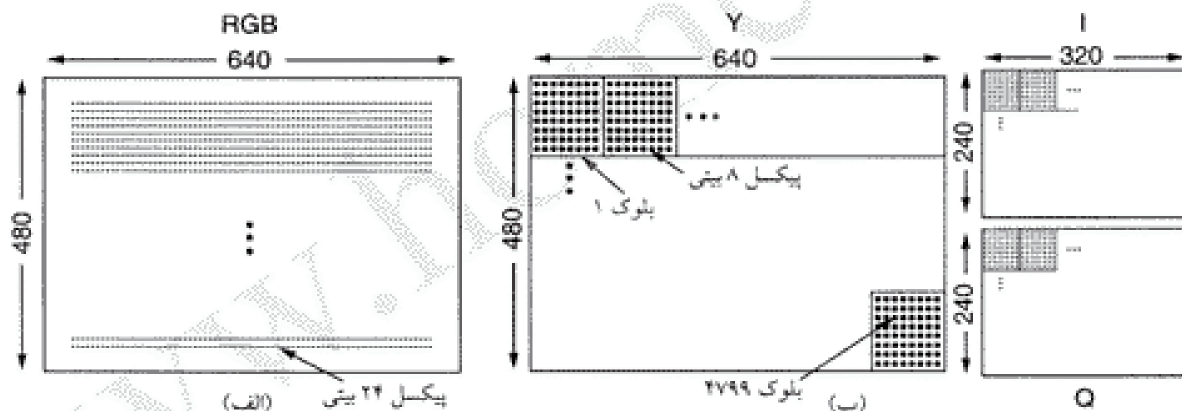
الگوریتم فشرده‌سازی). اما آنچه که ما به آن علاقه داریم (و در شکل ۷-۷۱می‌بینید)، **حالت متوالی تلفات‌دار** (lossy seqential) است. در اینجا برای سادگی بیشتر بحث، فقط به روش کُد کردن تصاویر ویدئویی 24-bit RGB در JPEG توجه می‌کنیم، و جزئیات دیگر را نادیده می‌گیریم.



شکل ۷-۷۱. عملکرد JPEG در حالت متوالی تلفات‌دار.

مرحلهٔ اول کُد کردن تصویر با JPEG ، **آماده‌سازی بلوک** (block preparation) است. برای سادگی بحث فرض می‌کنیم که ورودی JPEG یک تصویر RGB 480 × 640 با وضوح 24 bits/pixel است (شکل ۷-۷۲ الف). از آنجائیکه فشرده‌سازی روشنایی و رنگ نتایج بهتری دارد، ابتدا سیگنال روشنایی، Y ، و سیگنالهای رنگ، I و Q ، تصویر را (برای سیستم NTSC) با استفاده از فرمولهای زیر محاسبه می‌کنیم:

$$Y = 0.30R + 0.59G + 0.11B$$
$$I = 0.60R - 0.28G - 0.32B$$
$$Q = 0.21R - 0.52G + 0.31B$$



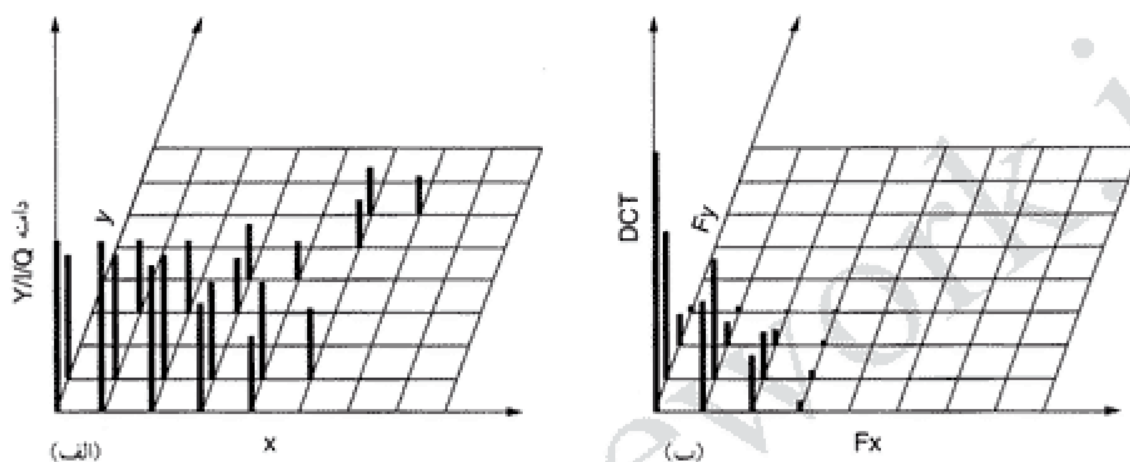
شکل ۷-۷۲. (الف) ورودی RGB . (ب) بعد از آماده‌سازی بلوک.

در PAL سیگنالهای رنگ V و U نامیده می‌شوند و ضرایب متفاوتی دارند، ولی ایدهٔ اصلی همان است؛ SECAM هم که با هر دوی آنها فرق دارد.

برای Y ، I و Q ماتریسهای جداگانه‌ای (با مقادیر 0 تا 255) ایجاد می‌شود. سپس، تصویر به بلوکهای مربعی ۴ پیکسلی تقسیم شده، و I و Q متوسط آنها محاسبه می‌شود (با این کار تصویر ورودی بـه 240 × 320 تبدیل می‌شود). این کاهش با تلفات همراه است، اما از آنجائیکه چشم انسان به روشنایی حساس‌تر است تا رنگ، متوجه آن نخواهد شد. ضریب کاهش تا اینجا ۱ به ۲ است (یعنی اندازهٔ فایل نصف شده است). حال، از تمام عناصر هر سه ماتریس عدد ۱۲۸ کم می‌شود، که با اینکار 0 به عدد میانه در محدودهٔ اعداد ماتریس‌ها تبدیل خواهد شد. و در آخر، تمام ماتریس‌ها به بلوکهای 8 × 8 تقسیم می‌شوند. با این کار ماتریس Y دارای 4800 بلوک، و دو ماتریس دیگر هر یک دارای 1200 بلوک خواهند بود (شکل ۷-۷۲ب را ببینید).

در مرحلهٔ دوم JPEG ، روی تمام 7200 ماتریس بصورت جداگانه **تبدیل کسینوسی گسسته** (Discrete – DCT

Cosine Transformation) انجام می‌شود. خروجی هر DCT یک ماتریس 8 × 8 از ضرایب DCT است، که عنصر (0, 0) هر تبدیل DCT مقدار متوسط آن بلوک است. عناصر دیگر نشان می‌دهند که در هر فرکانس فضایی چه مقدار انرژی طیفی وجود دارد. از نظر تئوری، DCT یک الگوریتم بدون‌تلفات است، ولی گرد شدن اعداد در محاسبات اعشاری و مثلثاتی عملاً باعث مقداری اتلاف اطلاعات خواهد شد. معمولاً مقدار عناصر ماتریس با دور شدن از مبدأ مختصات (0, 0) بسرعت تحلیل می‌روند (به شکل ۷-۷۳ نگاه کنید).



شکل ۷-۷۳. (الف) یکی از بلوکهای ماتریس Y . (ب) ضرایب DCT .

بعد از پایان DCT ، JPEG وارد مرحلهٔ سوم یعنی **کوانتیزه‌کردن** (quantization) می‌شود، که در آن ضرایب کم‌اهمیت‌تر DCT دور انداخته می‌شوند. در این تبدیل تلفات‌دار، عناصر ماتریس 8 × 8 ضرایب DCT بر وزنی که از یک جدول (جدول کوانتیزه کردن) گرفته می‌شود، تقسیم می‌شوند. اگر تمام وزن‌ها 1 باشند، تبدیل هیچ کاری انجام نمی‌دهد؛ ولی اگر وزن‌ها خیلی از مبدأ دور باشند، فرکانسهای فضایی بالاتر دور انداخته می‌شوند.

به مثال شکل ۷-۷۴ نگاه کنید. در این شکل سه جدول می‌بینید: ماتریس DCT اولیه، جدول کوانتیزه کردن، وجدولی که از تقسیم عناصر ماتریس DCT بر عناصر متناظر در جدول کوانتیزه کردن بدست آمده است. مقادیر جدول کوانتیزه کردن جزیی از استاندارد JPEG نیست، و هر برنامهٔ تبدیل به فرمت JPEG باید خود آنرا داشته باشد (و بوسیلهٔ آن مقدار تلفات اطلاعات را کنترل کند).

ضرایب DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

جدول کوانتیزه‌کردن

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

ضرایب کوانتیزه‌شده

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

شکل ۷-۷۴. محاسبهٔ ضرایب کوانتیزهٔ DCT .

در مرحلهٔ چهارم، بجای عنصر (0, 0) هر بلوک (گوشهٔ چپ-بالا) تفاضل آن با همین عنصر از جدول قبل نوشته می‌شود. از آنجائیکه این عناصر مقدار متوسط هر بلوک هستند، اختلاف آنها زیاد نیست و عددی که بدست

می‌آید نسبتاً کوچک است. فقط روی این عنصر است که تفاضل محاسبه می‌شود، نه عناصر دیگر. به عنصر (0, 0) هر جدول مؤلفهٔ DC (و به سایر عناصر، مؤلفهٔ AC) گفته می‌شود.

در مرحلهٔ پنجم تمام 64 عنصر جدول با روش کُدکردن run-length خطی می‌شوند. برای این کار از اسکن زیگزاگی جدول استفاده می‌شود (شکل ۷-۷۵)، چون در اسکن افقی-عمودی 0 ها کنار هم قرار نمی‌گیرند. در مثال شکل ۷-۷۵، اسکن زیگزاگی باعث شده تا ۳۸ صفر متوالی در انتهای ماتریس بدست آید. این ۳۸ عدد را می‌توان بطور خلاصه «۳۸ صفر» نامید (و جدول را خیلی کوچک کرد).

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

شکل ۷-۷۵. روش خطی کردن عناصر ماتریس کوانتیزه شده.

پس از اجرای مرحلهٔ پنجم، لیستی از اعداد (در فضای تبدیل) بدست می‌آید که نمایندهٔ تصویر فشرده شده هستند. در مرحلهٔ آخر برای کوچک کردن هر چه بیشتر این لیست، به اعدادی که بیشتر تکرار شده‌اند کُدهای کوچکتر داده می‌شود، و به اعداد با تکرار کمتر کُدهای بلندتر (**کُدهافمن**).

شاید JPEG پیچیده بنظر برسد، چون در واقع پیچیده هم هست. اما از آنجائیکه ضریب فشرده‌سازی در آن بسیار بالاست (نزدیک 20:1)، کاربرد گسترده‌ای پیدا کرده است. برای دیکُد کردن تصاویر JPEG ، الگوریتم بالا بصورت معکوس انجام می‌شود. JPEG تا حد زیادی متقارن است: دیکُد کردن تقریباً به همان اندازهٔ کُد کردن وقت می‌برد. اما همانطور که خواهید دید، کُدهایی هم هستند که بشدت نامتقارنند.

استاندارد MPEG

بالاخره به اصل مطلب رسیدیم: استانداردهای MPEG (Motion Picture Experts Group) . این استانداردها از سال ۱۹۹۳ بصورت بین‌المللی برای فشرده‌کردن ویدئو بکار برده می‌شوند، چون می‌توانند صدا و تصویر را با هم فشرده کنند (و ویدئو هم ترکیبی است از صدا و تصویر). تا اینجا فشرده‌سازی صدا و تصویر ثابت را دیدید. پس اجازه دهید فشرده‌سازی ویدئو را بررسی کنیم.

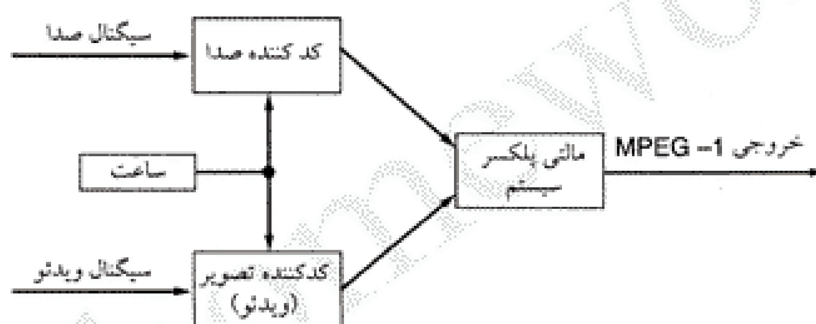
اولین استانداردی که نهایی شد، MPEG-1 بود (ISO 11172) . هدف این استاندارد تولید خروجی با کیفیت ضبط ویدئو (وضوح 240 × 352 در سیستم NTSC) با نرخ انتقال 1.2 Mbps بود. ویدئوی 240 × 352 با رنگ 24 bit/pixel و سرعت 25 frame/sec به پهنای باند 50.7 Mbps نیاز دارد، پس تقلیل آن به 1.2 Mbps یعنی فشرده‌سازی 40:1 ، و این اصلاً کار ساده‌ای نیست. از MPEG-1 برای انتقال ویدئو روی کابل زوج تابیده (در مسافتهای نه چندان زیاد)، و ذخیره کردن فیلم روی CD-ROM استفاده می‌شود.

استاندارد بعدی این خانواده MPEG-2 بود (ISO 13818)، که برای فشرده‌سازی دیدئو با کیفیت پخش به

پهنای باند 4-6 Mbps (معادل پهنای باند پخش NTSC یا PAL) طراحی شده بود. بعدها وضوح تصویر در MPEG-2 افزایش یافت، و HDTV را هم در بر گرفت. این فرمت امروزه بسیار رواج دارد، و از آن در DVD و تلویزیون ماهواره‌ای دیجیتال استفاده می‌شود.

اصول کار در MPEG-1 و MPEG-2 یکسان است، و آنها فقط در جزئیات با هم فرق دارند. در واقع MPEG-2 همان MPEG-1 است، که ویژگیهای دیگری (از قبیل فرمت‌ها و روشهای کُد کردن) به آن اضافه شده است. ما هم ابتدا MPEG-1 و سپس MPEG-2 را بررسی خواهیم کرد.

MPEG-1 سه قسمت دارد: صدا، ویدئو، و سیستم که دو قسمت قبلی را یکپارچه می‌کند (شکل ۷-۷۶ را ببینید). صدا و ویدئو بصورت جداگانه و مستقل کُد می‌شوند، که این کار مشکل سنکرون کردن آنها در گیرنده را پیش می‌آورد. این مشکل با استفاده از یک ساعت سیستم 90-kHz (که وقت فعلی را به هر دو قسمت صدا و ویدئو می‌دهد) حل شده است. اینها اعداد ۳۳ بیتی هستند، بنابراین یک فیلم می‌تواند بدون هیچ مشکلی حتی ۲۴ ساعت طول بکشد (بدون اینکه مسئلهٔ صفر شدن تایمر پیش آید). این برچسبهای زمانی در خروجی کُدشده هم نوشته می‌شوند، و گیرنده می‌تواند از آنها برای سنکرون کردن صدا و ویدئو استفاده کند.



شکل ۷-۷۶. سنکرون کردن صدا و ویدئو در MPEG-1 .

اکنون اجازه دهید فشرده‌سازی ویدئو در MPEG-1 را بررسی کنیم. در هر فیلم دو نوع افزونگی وجود دارد: فضایی و موقتی؛ MPEG-1 از هر دوی آنها استفاده می‌کند. برای بکارگیری افزونگی فضایی، هر فریم بطور جداگانه با JPEG کُد می‌شود. این رهیافت بیشتر در مواقعی بکار برده می‌شود که (علاوه بر پخش فیلم) به تک تک فریمها نیز (برای کارهایی مانند تدوین فیلم) احتیاج داشته باشیم. با این روش می‌توان به پهنای باند 8-10 Mbps دست یافت.

برای رسیدن به فشردگی بیشتر می‌توان از این واقعیت که در یک فیلم فریمهای متوالی تقریباً یکسان هستند، بهره گرفت. البته مقدار کاهش حاصله از این ویژگی آنچنان که در نگاه اول بنظر می‌رسد زیاد نیست، چون در اغلب فیلمها هر ۳ یا ۴ ثانیه (بطور متوسط ۷۵ فریم) صحنه بکلی عوض می‌شود. اما همین توالیهای ۷۵ فریمی تقریباً یکسان هم به کاهش قابل‌ملاحظه‌ای (در مقایسه با JPEG) منجر خواهد شد.

در صحنه‌هایی که زمینه یا دوربین ثابت است و هنرپیشه‌ها حرکت کمی دارند، تقریباً تمام پیکسلها در فریمهای متوالی شبیه هم هستند. در این حالت، محاسبهٔ تفاضل دو فریم و اجرای الگوریتم JPEG روی این تفاضل، بخوبی کار خواهد کرد. اما در صحنه‌هایی که حرکت دوربین زیاد است، این تکنیک به هیچ دردی نخواهد خورد، و باید راهی پیدا کرد که حرکت شدید صحنه را جبران کند. این دقیقاً همان کاری است که MPEG انجام می‌دهد؛ و تفاوت MPEG و JPEG نیز در همین جاست.

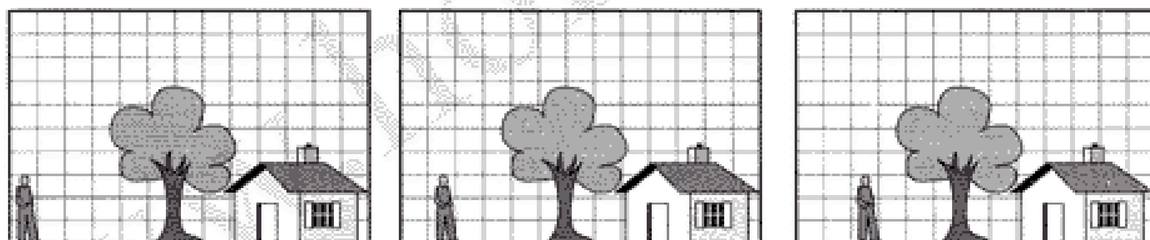
در خروجی MPEG-1 چهار نوع فریم می‌تواند وجود داشته باشد:

۱. فریمهای I (Intracoded): تصاویر ثابت و مستقل JPEG .
۲. فریمهای P (Predictive): تفاوت بلوک-به-بلوک با فریم قبلی.
۳. فریمهای B (Bidirectional): تفاوتهای بین فریم قبلی و بعدی.
۴. فریمهای D (DC-coded): متوسط بلوک‌ها برای جلو رفتن سریع فیلم (FF) .

فریمهای I تصاویر ثابت با فرمت JPEG هستند، که همچنین از وضوح کامل روشنایی و وضوح نیمه کامل در هر محور رنگ استفاده می‌کنند. سه دلیل برای قرار دادن فریمهای I در استریم خروجی وجود دارد. اول اینکه، از MPEG-1 در سیستمهای چندپخشی (multicast) ، که تعداد زیادی بینندهٔ مستقل دارند، نیز استفاده می‌شود. اگر هر فریم به فریم قبلی (همینطور تا اولین فریم فیلم) وابسته باشد، کسی که وسط فیلم تلویزیون خود را روشن کرده (و اولین فریم را از دست داده)، هیچ چیز نمی‌تواند ببیند (چون در واقع مبنایی برای دیکُد کردن فریمها ندارد). دوم، اگر یکی از فریمها خراب شود، دیگر نمی‌توان فریمهای بعدی را دیکُد کرد. سوم اینکه، بدون فریمهای I کار گیرنده برای جلو یا عقب رفتن سریع (FF یا Rewind) بسیار مشکل خواهد شد، چون مجبور است تک تک فریمها را دیکُد کند. به این دلایل، در هر ثانیه یک یا دو فریم I در خروجی قرار داده می‌شود.

بر خلاف فریمهای I ، فریمهای P فقط اختلاف بین فریمها را کُد می‌کنند. فریمهای P از ایدهٔ **ماکروبلوک** (macroblock) - ماتریسهای 16×16 پیکسل در فضای روشنایی، و ماتریسهای 8×8 پیکسل در فضای رنگ - استفاده می‌کنند. هر ماکروبلوک با جستجوی بلوک مشابه در فریم قبلی (برای یافتن تشابه یا اختلاف) کُد می‌شود.

شکل ۷-۷۷ نمونه‌ای که در آن فریمهای P بکار می‌آیند، را نشان می‌دهد. در اینجا سه فریم متوالی می‌بینید که زمینهٔ صحنه در آنها یکسان است، و فقط جای هنرپیشه عوض شده است. در این فریمها، ماکروبلوک‌های زمینهٔ صحنه دقیقاً یکسان مانده، و فقط ماکروبلوک‌های مربوط به هنرپیشه عوض شده و باید کُد شود.



شکل ۷-۷۷. سه فریم متوالی.

استاندارد MPEG-1 هیچ حرفی دربارهٔ روش جستجو، عمق جستجو، و یا مفهوم یکسان بودن، نمی‌زند - تمام اینها برعهدهٔ نویسندهٔ برنامه گذاشته شده است. برای مثال، در یک برنامه ممکنست جستجوی ماکروبلوک در مکان فعلی در فریم قبلی، و در تمام آفست‌های $\pm\Delta x$ (در جهت x) و $\pm\Delta y$ (در جهت y) صورت گیرد، و تعداد نقاط یکسان از نظر روشنایی محاسبه شود. مکانی با مقدار بیشتر (مشروط باینکه از یک آستانهٔ تعریف شده بالاتر باشد) بعنوان برنده انتخاب می‌شود. در غیراینصورت، گفته می‌شود که ماکروبلوک گم شده است. البته الگوریتمهای بسیار بهتری نیز برای این منظور وجود دارد.

اگر ماکروبلوک پیدا شود، اختلاف (روشنایی و رنگ) آن با فریم قبلی محاسبه شده، و سپس با الگوریتم JPEG (تبدیل DCT ، کوانتیزه کردن، کُد run-length ، و کُد هافمن) کُد می‌شود. مقدار این ماکروبلوک در استریم خروجی بصورت بردار حرکت آن (شامل مقدار و جهت جابجایی) ثبت می‌شود. اگر یک ماکرو بلوک در فریم قبلی وجود نداشته باشد، بصورت عادی (فریم I) کُد می‌شود.

این الگوریتم بشدت نامتقارن است. برنامه آزاد است هر مکان قابل‌قبولی در فریم قبلی را که بخواهد (برای

یافتن ماکروبلوک موردنظر) امتحان کند. با این روش استریم MPEG-1 می‌تواند به ضریب فشرده‌سازی بالایی دست پیدا کند، ولی در ضمن زمان کُد کردن هم بالا خواهد رفت. همانطور که می‌توان حدس زد، این رهیافت برای کُد کردن فیلمهای کتابخانه‌ای مناسب است، ولی اصلاً بدرد ویدئوکنفرانس نمی‌خورد.

برنامه‌نویس در اتخاذ تصمیم برای تعریف مفهوم «ماکروبلوک یکسان» نیز آزاد است. این آزادی دست برنامه‌نویس را برای انتخاب نقطهٔ تعادل بین سرعت و کیفیت (در عین اطمینان از اینکه خروجی همواره با MPEG-1 سازگار است) باز می‌گذارد. در هر حال، خروجی یا ماکروبلوک JPEG شده است، یا JPEG شدهٔ اختلاف آن با فریم قبلی.

تا اینجا، دیکُد کردن MPEG-1 ساده است. دیکُد کردن فریمهای I که هیچ فرقی با تصاویر ثابت JPEG ندارد. برای دیکُد کردن فریمهای P ، برنامه فریم قبلی را در یک بافر ذخیره کرده و در بافر دیگر فریم جدیدی بر اساس ماکروبلوکهای کامل و ماکروبلوکهای اختلافها با فریم قبلی می‌سازد. فریم جدید بلوک به بلوک ساخته می‌شود.

فریمهای B شبیه فریمهای P هستند، با این تفاوت که فریم مبنای آنها می‌تواند (بجای فریم قبلی) فریم بعدی نیز باشد. این آزادی کمک زیادی به جبران حرکت صحنه‌ها می‌کند، بویژه وقتی اشیاء روی صحنه از جلو یا پشت اشیاء دیگر عبور می‌کنند. برای دیکُد کردن فریمهای B ، برنامه باید سه فریم را در آن واحد در حافظه داشته باشد: فریم قبلی، فریم فعلی، و فریم بعدی. با آنکه فریمهای B ضریب فشرده‌سازی را بالا می‌برند، همهٔ برنامه‌های MPEG از آنها پشتیبانی نمی‌کنند.

فریمهای D فقط برای نمایش تصویری با وضوح پائین هنگام جلو یا عقب رفتن سریع فیلم (FF یا Rewind) بکار برده می‌شوند. دیکُد کردن MPEG-1 با سرعت پخش معمولی باندازهٔ کافی دشوار هست، انجام همین کار با سرعت ۱۰ برابر که دیگر جای خود دارد. بهمین دلیل در سرعتهای بالا از فریمهای D (که وضوح کمتری دارند) استفاده می‌شود. هر فریم D در واقع فقط مقدار متوسط هر بلوک (بدون هر گونه کُد کردن اضافی) است، که نمایش آنرا در زمان واقعی آسان می‌کند. با این ویژگی می‌توان یک فیلم را با سرعت زیاد بدنبال صحنهٔ موردنظر جستجو کرد. فریمهای D معمولاً درست قبل از فریمهای I قرار داده می‌شوند، تا وقتی حرکت سریع فیلم متوقف شد، بتوان نمایش را با سرعت معمولی ادامه داد.

اجازه دهید بعد از MPEG-1 ، سراغ MPEG-2 برویم. روش کُد کردن MPEG-2 اساساً شبیه MPEG-1 است، با این تفاوت که MPEG-2 از فریمهای D پشتیبانی نمی‌کند. همچنین، در DCT بجای ماتریسهای 8×8 از بلوکهای 10×10 استفاده می‌کند، که با این کار تعداد ضرایب ۵۰ درصد بیشتر شده و کیفیت بالاتر می‌رود. از آنجائیکه MPEG-2 برای پخش تلویزیونی و DVD طراحی شده، از تصاویر خط-در-میانی و پیشرونده پشتیبانی می‌کند (در حالیکه MPEG-1 فقط از تصاویر پیشرونده پشتیبانی می‌کند). بین این دو استاندارد تفاوتهای کوچک دیگری نیز وجود دارد.

بجای یک وضوح ثابت، MPEG-2 از چهار وضوح پشتیبانی می‌کند: کم (352×240)، اصلی (720×480)، بالا-۱۴۴۰ (1440×1152)، و بالا (1920×1080). وضوح 352×240 برای دستگاههای VCR (و سازگاری با MPEG-1) در نظر گرفته شده است. وضوح اصلی برای پخش NTSC بکار می‌رود. دو وضوح دیگر برای HDTV در نظر گرفته شده‌اند. MPEG-2 در وضوحهای بالا معمولاً با سرعت 4-8 Mbps اجرا می‌شود.

۷-۴-۸ پخش فیلم بر حسب تقاضا

پخش فیلم بر حسب تقاضا (Video on demand - VOD) چیزی شبیه مغازهٔ کرایهٔ فیلمهای ویدئویی است. در آنجا مشتری یکی از فیلمهای موجود را انتخاب کرده و برای تماشا به منزل می‌برد. ولی در اینجا نیازی به مراجعه به

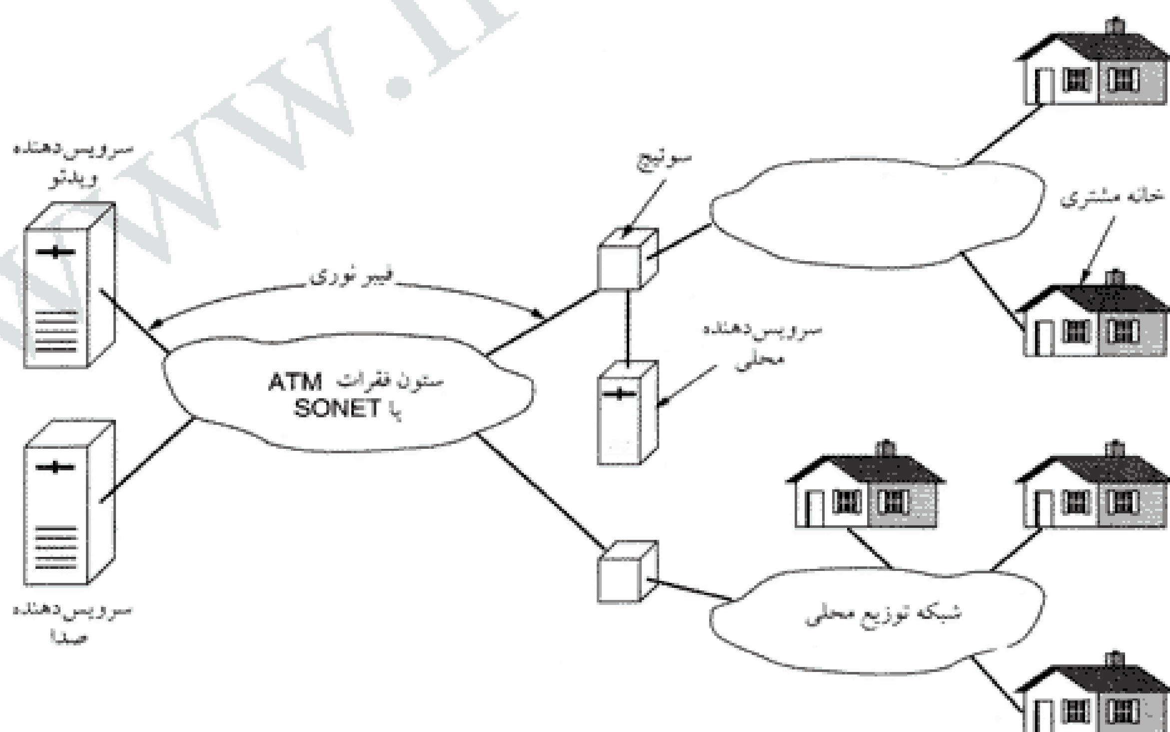
مغازه نیست و انتخاب فیلم از طریق دستگاه کنترل از راه دور تلویزیون انجام شده، و پخش آن هم بلافاصله شروع می‌شود. لازم به گفتن نیست که، پیاده‌سازی VOD از تعریف آن کمی مشکلتر است.

آیا پخش فیلم بر حسب تقاضا شبیه کرایهٔ فیلم ویدئویی است، یا انتخاب کانال در تلویزیونهای کابلی؟ پاسخ این سئوال تبعات فنی مهمی دارد. وقتی یک فیلم ویدئویی کرایه می‌کنید، می‌توانید وسط فیلم آنرا متوقف کرده، و بعد از خوردن یک فنجان چای یا جواب دادن به تلفن، ادامهٔ آنرا از همانجایی که متوقف شده بود، تماشا کنید. اما بینندگان کانالهای تلویزیونی نمی‌توانند چنین کاری بکنند.

اگر VOD بخواهد رقابت مؤثری با مغازه‌های کرایهٔ فیلم داشته باشد، باید به مشتری اجازه دهد فیلم را بدلخواه خود متوقف کرده، و یا عقب و جلو ببرد. چنین کاری مستلزم آن است که برای هر مشتری یک کپی اختصاصی از فیلم پخش شود.

اما اگر VOD را فقط نوعی تلویزیون پیشرفته فرض کنیم، آنگاه کافیست فیلمهای پُرطرفدار را در فواصل ۱۰ دقیقه‌ای (و بدون توقف تا آخر) پخش کنیم. در این حالت، مشتری برای دیدن این فیلم فقط کافیست (حداکثر) ۱۰ دقیقه صبر کند. با اینکه متوقف کردن چنین فیلمی وجود ندارد، اما اگر وسط آن کاری برایتان پیش آمد، کافیست به کانال دیگری (که ۱۰ دقیقه عقبتر است) رفته و فیلم را از جایی که از دست داده بودید، تماشا کنید. (تکرار چیز خوبی نیست، ولی مسلماً بهتر از ندیدن قسمتهایی از فیلم است.) این روش را «تقریباً VOD» می‌نامند. هزینهٔ پیاده‌سازی چنین سیستمی بسیار کمتر است، چون می‌توان هر فیلم را برای عدهٔ زیادی پخش کرد. تفاوت VOD و «تقریباً VOD» مثل مسافرت با اتومبیل شخصی و اتوبوس است.

تماشای فیلم با VOD (یا «تقریباً VOD») یکی از امکانات بالقوهٔ شبکه‌های پهن‌باند امروزی است، که در شکل ۷-۷۸ یکی از مدلهای متداول آنرا می‌بینید. در مرکز این سیستم یک شبکهٔ با پهنای باند زیاد (ملی یا بین‌المللی) قرار دارد، که هزاران شرکت توزیع‌کننده (شرکتهای تلفن، تلویزیون کابلی و مانند آنها) به آن متصلند.



شکل ۷-۷۸. یک سیستم پخش فیلم بر حسب تقاضا (VOD) .

انشعابات این سیستم از طرف دیگر به هزاران خانه وارد شده، و در آنجا به ترمینالهای هوشمند (که در واقع کامپیوترهای تخصصی هستند) متصل می‌شود.

تعداد زیادی شرکت سرویس‌دهنده نیز با فیبرهای نوری پُرظرفیت به ستون‌فقرات شبکه وصل هستند، که برخی از آنها سرویسهای عمومی مانند **«پول‌بده-تماشاکن»** (pay-per-view) یا **«پول‌بده-گوش‌کن»** (pay-per-hear) ، یا سرویسهای تخصصی مانند خریدازخانه (home shopping) ارائه می‌کنند. چنین شبکه‌ای می‌تواند سرویسهایی مانند اخبار، ورزش، فیلم و سریال، دسترسی وب، و هزاران امکان بالقوهٔ دیگر در اختیار کاربران قرار دهد.

در این سیستم سرویس‌دهنده‌های کوچکتری در نزدیکی مشتریان تعبیه می‌شود (local spooling server) ، که ویدئوهای درخواستی را ذخیره می‌کنند تا ترافیک شبکه در ساعات اوج مصرف کمتر شود. این کارها چگونه باید انجام شوند و چه کسی باید آنها را انجام دهد؟ هنوز بحث‌های سختی در جریان است. در اینجا ما فقط به قسمتهای اصلی سیستم می‌پردازیم: **سرویس‌دهنده‌های ویدئو** (video server) و **شبکهٔ توزیع** (distribution network) .

سرویس‌دهنده‌های ویدئو

برای ایجاد یک سیستم VOD (یا «تقریباً VOD») به سرویس‌دهنده‌های ویدئو، که بتوانند تعداد زیادی فیلم سینمایی را ذخیره و همزمان پخش کنند، نیاز داریم. تعداد کل فیلمهای سینمایی که تاکنون ساخته شده، چیزی در حدود ۶۵,۰۰۰ تخمین زده شده است (Minoli, 1995) . یک فیلم معمولی با فرمت MPEG-2 چیزی در حدود 4 GB حجم خواهد داشت، بنابراین برای ذخیره کردن ۶۵,۰۰۰ فیلم به 260 TB (TB = ترابایت) نیاز داریم. اگر فیلمها و سریالهای قدیمی تلویزیونی، فیلمهای خبری و ورزشی، و کاتالوگهای ویدئویی را هم به آن اضافه کنیم، آن وقت متوجه می‌شوید که مشکل کجاست!

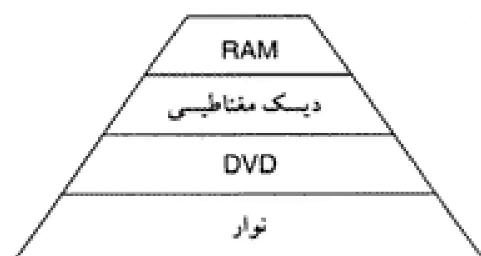
ارزانترین وسیلهٔ برای ذخیره کردن حجم زیادی از اطلاعات، نوار مغناطیسی است - و بنظر می‌رسد در آیندهٔ نزدیک وسیلهٔ ارزانتری به بازار نخواهد آمد. روی یک نوار مغناطیسی 200-GB می‌توان ۵۰ فیلم MPEG-2 (با هزینهٔ ۱ تا ۲ دلار برای هر فیلم) ذخیره کرد. امروزه سرویس‌دهنده‌های بزرگ ویدئویی که می‌توانند هزاران نوار مغناطیسی را در خود نگه دارند، و برای جابجا کردن فیلمها به بازوهای روباتیک مجهز هستند، به بازار آمده‌اند. مشکل این سیستمها زمان دسترسی به فیلمها (بخصوص فیلم پنجاهم)، سرعت انتقال، و محدودیت تعداد دستگاههای پخش است (برای پخش همزمان n فیلم، به n دستگاه پخش نیاز داریم).

خوشبختانه، تجربهٔ مغازه‌های کرایهٔ ویدئو، کتابخانه‌های عمومی، و سازمانهای دیگر نشان می‌دهد که تمام آیتمها دارای محبوبیت یکسانی نیستند. طبق یک فرمول تجربی، اگر N فیلم داشته باشیم، احتمال درخواست برای k امین فیلم محبوب تقریباً C/k است - که در آن C بصورت زیر محاسبه می‌شود:

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdots + 1/N)$$

طبق این فرمول (که به قانون زیف - Zipf law - معروف است) محبوبترین فیلم هفت برابر فیلم هفتم طرفدار دارد (Zipf, 1994) . با استفاده از این واقعیت می‌توان به یک مدل ذخیره‌سازی سلسله مراتبی دست یافت (شکل ۷-۷۹ را ببینید). در این مدل، بالا رفتن در هرم باعث افزایش کارایی (سرعت دسترسی، و سرعت انتقال) می‌شود.

گزینهٔ بعدی برای ذخیره‌سازی فیلمهای ویدئویی، دیسک نوری است. دیسکهای DVD در حال حاضر ظرفیتی معادل 4.7 GB دارند، که برای ذخیره کردن یک فیلم MPEG-2 کافیست، ولی نسل بعدی DVD برای ۲ فیلم جای کافی خواهد داشت. با آن که زمان جستجو در دیسکهای نوری در مقایسه با نوار مغناطیسی بیشتر است



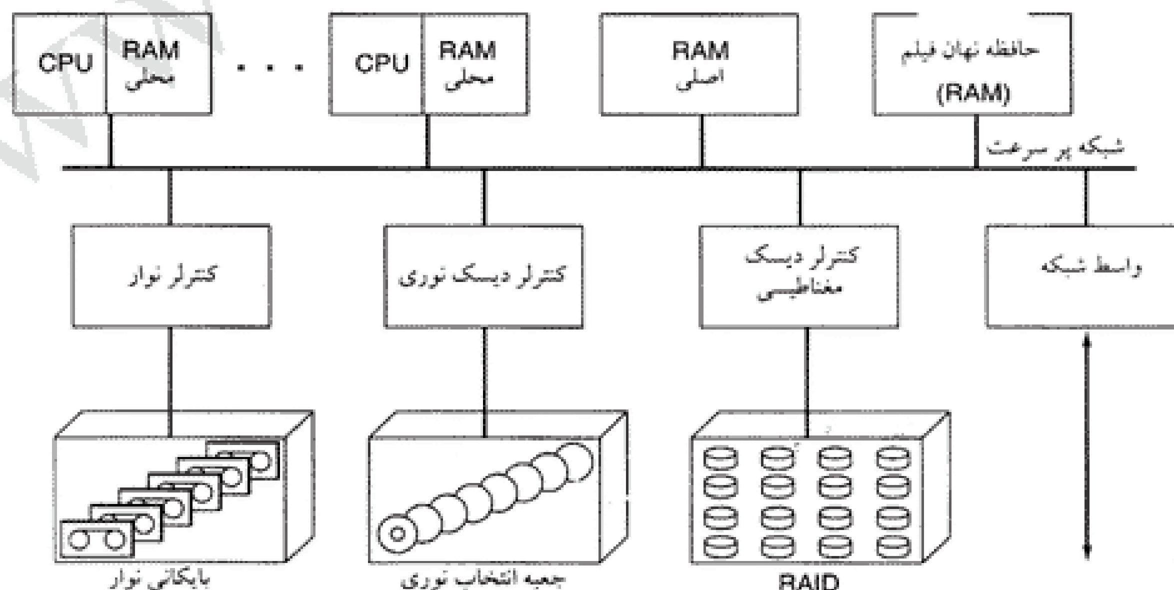
شکل ۷-۷۹. سلسله مراتب ذخیره‌سازی در سرویس‌دهنده‌های ویدئو.

(50 msec در مقابل 5 msec)، ولی آنها ارزانتر و مقاومتر هستند، و بزودی شاهد سرویس‌دهنده‌های ویدئو با ظرفیت هزاران DVD در بازار خواهیم بود.

در مرحلهٔ بعد دیسکهای مغناطیسی قرار دارند. این دیسکها با زمان دسترسی پائین (5 msec)، نرخ انتقال بالا (320 MB/sec در دیسکهای SCSI) و ظرفیت زیاد (> 100 MB) گزینهٔ خوبی برای ذخیره کردن فیلمهای پُربیننده هستند. عیب عمدهٔ این دیسکها قیمت بالای آنهاست.

در بالاترین نقطهٔ هرم شکل ۷-۷۹ حافظهٔ RAM قرار دارد. با اینکه قیمت RAM در سالهای اخیر بشدت کاهش پیدا کرده، ولی ذخیره کردن یک فیلم MPEG-2 به ۲۰۰ دلار حافظهٔ RAM نیاز دارد - و برای ذخیره کردن فقط ۱۰۰ فیلم باید ۲۰,۰۰۰ دلار هزینه کنیم (که البته برای یک سرویس‌دهندهٔ ویدئوی سطح بالا چندان زیاد و غیرعملی نیست).

از آنجائیکه یک سرویس‌دهندهٔ ویدئو اساساً دستگاهیست برای I/O بی‌درنگ در حجمهای بالا، سخت‌افزار و نرم‌افزار آن بایستی تفاوت اساسی با کامپیوترهای ویندوز و یونیکس معمولی داشته باشد. در شکل ۷-۸۰ معماری سخت‌افزاری یک سرویس‌دهندهٔ ویدئوی نوعی را می‌بینید. قسمتهای مختلف این سرویس‌دهنده عبارتند از: یک یا چند CPU سریع (با مقداری RAM اختصاصی برای هر کدام)، یک حافظهٔ اصلی مشترک، یک حافظهٔ نهان بزرگ از نوع RAM برای فیلمهای پُربیننده، ترکیبی از وسایل ذخیره‌سازی متنوع برای ذخیره کردن فیلمها، و سخت‌افزار شبکه (معمولاً ارتباط فیبر نوری به ستون فقرات ATM یا SONET با سرعت OC-12 یا بالاتر) - که این قسمتها با یک باس فوق سریع (حداقل 1 GB/sec) به یکدیگر متصل می‌شوند.



شکل ۷-۸۰ معماری سخت‌افزاری یک سرویس‌دهندهٔ ویدئو.

نرم‌افزار سرویس‌دهندهٔ ویدئو خود داستان دیگریست. در این سیستم، وظیفهٔ CPU ها عبارتست از: گرفتن درخواست مشتریان، پیدا کردن فیلمها، منتقل کردن فیلم بین قسمتهای مختلف، نگهداری صورتحساب مشتریان، و مانند آن. زمان در برخی از این کارها نقش حیاتی دارد، بهمین دلیل باید از سیستم عامل بی‌درنگ استفاده کنیم (البته نه در همهٔ CPUها). در این سیستمها، هر وظیفه به بخشهای کوچکتر تقسیم می‌شود، و هر بخش باید در زمان معین به پایان برسد. برای زمانبندی این وظایف می‌توان از الگوریتمهایی مانند **«نزدیکترین بن‌بست‌درنوبت‌بعد»** (nearest deadline next) یا **«نرخ یکنواخت»** (rate monotonic) استفاده کرد (Liu and Layland, 1973) .

این نرم‌افزار همچنین خصلت واسط سمت مشتری (سرویس‌دهنده‌های بینابینی و گیرنده) را نیز تعیین می‌کند. دو نوع واسط کاربر بیشتر رواج دارند. اولی یک سیستم فایل معمولی است، که مشتری می‌تواند فایلهای موردنظرش را باز کرده، بخواند، بنویسد، و سپس ببندد. چنین واسطی می‌تواند سیستم فایلی شبیه یونیکس داشته باشد (البته با در نظر داشتن مشکلاتی که از ساختار سلسله مراتبی سیستم و بی‌درنگ بودن آن ناشی می‌شود).

واسط کاربر دوم به دستگاههای ضبط ویدئو شبیه است، با فرمانهایی مانند باز کردن فیلم، پخش، توقف، سریع به جلو، و سریع به عقب. تفاوت این واسط با قبلی (شبیه یونیکس) این است که به محض شروع پخش فیلم، سرویس‌دهنده (بدون نیاز به فرمان اضافی) داده‌ها را به سمت مشتری پمپ می‌کند.

قلب سرویس‌دهندهٔ ویدئو نرم‌افزار **«مدیریت‌دیسک»** (disk management) است. این نرم‌افزار دو وظیفهٔ اصلی دارد: انتقال فیلم از نوار مغناطیسی یا دیسک نوری به دیسک مغناطیسی، و انجام بموقع درخواست‌های خواندن دیسک. وظیفهٔ اول (یعنی انتقال فیلم) نقش مهمی در افزایش کارایی سیستم دارد.

برای سازماندهی دیسکها دو روش وجود دارد: **مزرعهٔ‌دیسک** (disk farm) ، و **آرایهٔ‌دیسک** (disk array) . در روش اول، مزرعهٔ دیسک، روی هر دیسک چند فیلم ذخیره می‌شود (که برای کارایی و اطمینان بیشتر می‌توان هر فیلم را روی چند دیسک - حداقل دو تا - ذخیره کرد). در روش دوم، آرایهٔ دیسک یا RAID (**آرایهٔ‌افزونه‌از دیسکهای‌ارزان** - Redundant Array of Inexpensive Disks)، هر فیلم روی چندین دیسک مختلف پخش می‌شود. به این روش - که در آن فیلم به n نوار باریک تقسیم شده، و این نوارها روی دیسکهای 0 تا $n - 1$ ذخیره می‌شود - **نوارکردن** (striping) گفته می‌گویند.

یک آرایهٔ دیسکهای نواری چندین مزیت نسبت به مزرعهٔ دیسک دارد. اول، همهٔ n دیسک را می‌توان بطور همزمان خواند یا نوشت، و این کارایی سیستم را n برابر می‌کند. دوم، می‌توان با قرار دادن یک دیسک اضافی در هر گروه n تایی از دیسکها، و نوشتن حاصل XOR تمام نوارهای اصلی روی این دیسک، آنها را در مقابل خرابی محافظت کرد. و بالاخره، متعادل کردن بار بطور خودکار انجام خواهد شد، و نیازی نیست فیلمهای پُربیننده را بصورت دستی بین دیسکهای مختلف پخش کنیم. از طرف دیگر، آرایهٔ دیسک تکنیکی پیچیده است، و اگر چندین دیسک با هم خراب شوند، دسترسی به کل فیلمها غیرممکن خواهد شد. پیاده‌سازی ویژگیهای «سریع به جلو» یا «سریع به عقب» (که جزء الزامات واسط نوع دوم هستند) نیز در این سیستم بسیار دشوار است.

وظیفهٔ دیگر نرم‌افزار مدیریت دیسک دادن سرویس بی‌درنگ به درخواستهای مشتریان است. تا چند سال پیش چنین وظیفه‌ای مستلزم طراحی الگوریتمهای پیچیدهٔ زمانبندی دیسک بود، ولی امروزه با کاهش شدید قیمت RAM روشهای بسیار ساده‌تری ممکن شده است. برای اینکه بتوان هر استریم را بصورت بی‌درنگ به مشتری فرستاد، برای هر فیلم بافری از ۱۰ ثانیه ویدئو (معادل 5 MB) در RAM نگه داشته می‌شود. این بافر توسط واسط دیسک پُر، و توسط واسط شبکه خالی می‌شود. با 500 MB RAM می‌توان ۱۰۰ استریم را به این شکل (مستقیماً از حافظه) سرویس داد. البته برای اینکه زیرسیستم دیسک بتواند این بافرها را بطور پیوسته تغذیه کند، باید سرعتی معادل 50 MB/sec داشته باشد، که این ویژگی با بکارگیری دیسکهای SCSI جدید بسادگی امکانپذیر است.

شبکهٔ توزیع

شبکهٔ توزیع (distribution network) عبارتست از مجموعهٔ سوئیچها و خطوط ارتباطی بین مبدأ و مقصد. همانطور که در شکل ۷-۷۸ دیدید، در این شبکه یک ستون فقرات شبکه وجود دارد، که به شبکهٔ توزیع محلی متصل است. معمولاً سوئیچینگ فقط در ستون فقرات وجود دارد، نه در شبکهٔ توزیع محلی.

مهمترین چیزی که ستون فقرات باید داشته باشد، پهنای باند زیاد است. پائین بودن میزان **لرزش** (jitter - وقفه‌های کوچک و ناخوشایندی که در اثر ترافیک زیاد در پخش صدا و تصویر رُخ می‌دهد) از دیگر الزامات ستون فقرات است، ولی از آنجائیکه ضعیفترین کامپیوترهای امروزی نیز می‌توانند حداقل ۱۰ ثانیه ویدئو با کیفیت MPEG-2 را بافر کنند، این ویژگی اهمیت سابق خود را از دست داده است.

بی‌نظمی عجیبی بر شبکه‌های محلی حکمفرماست، چون شرکتهای زیادی سعی می‌کنند تا خدمات متنوع خود را به مشتریان بفروشند. شرکتهای تلفن، تلویزیون کابلی (و اخیراً شرکتهای برق) متقاعد شده‌اند که برای برنده شدن در این میدان رقابت باید نفر اول باشند. در نتیجه، هر روز تکنولوژی جدیدی وارد بازار مصرف می‌شود. در ژاپن حتی شرکتهای فاضلاب وارد تجارت اینترنت شده‌اند، با این استدلال که گسترده‌ترین شبکهٔ خطوط لوله متعلق به آنهاست، و می‌توانند فیبرهای نوری را به هر خانه‌ای بکشند (فقط باید دقت کنند کابلهای آنها از کجا سر در می‌آورد!). چهار روش توزیع محلی VOD عبارتند از: ADSL ، FTTC ، FTTH ، و HFC - اجازه دهید آنها را بررسی کنیم.

شرکتهای تلفن اولین بار با تکنولوژی ADSL وارد میدان رقابت شبکه‌های توزیع محلی شدند. در فصل ۲ دربارهٔ تکنولوژی ADSL صحبت کردیم، و نیازی به تکرار آن نیست. ایدهٔ اصلی ADSL استفاده از سیمهای مسی است که تقریباً به هر خانه‌ای (در اروپا، آمریکا و ژاپن) کشیده شده‌اند. اگر می‌شد از این سیمها برای پخش فیلم استفاده کرد، که نان شرکتهای تلفن توی روغن بود! ولی مشکل اینجاست که سیمهای مسی در فواصل ۱۰ کیلومتری حتی برای پخش MPEG-1 مناسب نیستند، چه رسد به MPEG-2 . فیلمهای تمام رنگی با وضوح بالا به پهنای باند 4-8 Mbps (بسته به کیفیت موردنظر) نیاز دارند، و ADSL (جز در مسافتهای بسیار کوتاه) نمی‌تواند چنین سرعتی ارائه کند.

دومین طرح شرکتهای تلفن FTTC (**فیبرنوری تاکوچه** - Fiber To The Curb) است. در FTTC ، شرکت تلفن از ایستگاه پایانی (end office) یک رشته فیبر نوری به هر محله می‌کشد، و آنرا به دستگاهی موسوم به ONU (**واحدشبکهٔ نوری** - Optical Network Unit) وصل می‌کند. به هر ONU می‌توان تا ۱۶ زوج سیم مسی تلفن وصل کرد. با این تمهید، طول کابلهای مسی آنقدر کوتاه می‌شود که دستیابی به T1 یا T2 دو-طرفهٔ همزمان (که بترتیب برای MPEG-1 و MPEG-2 مناسبند) را ممکن می‌سازد. این سرویس (بعلت متقارن بودن) برای ویدئوکنفرانس نیز کاملاً مناسب است.

سومین راه حل شرکتهای تلفن کشیدن فیبر نوری تا در خانهٔ مشتریان است، که FTTH (**فیبرنوری تاخانه** - Fiber To The Home) نام دارد. در این طرح، هر فرد می‌تواند یک کاربر OC-1 ، OC-3 ، یا حتی بیــ خانه‌اش داشته باشد. FTTH بسیار گران است و بزودی عملی نخواهـ ـد، ولی می‌توان تصور کرد که ـ امکاناتی در اختیار مشتریان می‌گذارد. در شکل ۷-۶۳ دیدید که چگونه هر فردی می‌تواند یک ایستگاه رادیویی شخصی راه بیندازد؛ با داشتن FTTH راه‌اندازی ایستگاه تلویزیون شخصی چندان دور از ذهن نیست. هر سه طرح ADSL ، FTTC و FTTH روشهایی نقطه-به-نقطه هستند، و این از شرکتهای تلفن (که به این نوع ارتباطات عادت دارند) چندان بعید نیست.

رهیافت HFC (**آمیختهٔ فیبر-کواکس** - Hybrid Fiber Coax) که از سوی شرکتهای تلویزیو... ...ـه شده،

یکلی با سه روش قبلی متفاوت است (به شکل ۲-۴۷ الف نگاه کنید). جریان از این قرار است: شرکتهای تلویزیون کابلی در حال تعویض کابلهای کواکس 300-450 MHz (با ظرفیت ۵۰ تا ۷۵ کانال 6-MHz) با کابلهای کواکس 750-MHz (که ۱۲۵ کانال 6-MHz ظرفیت دارند) هستند، که فقط ۷۵ تا از این کانالها برای پخش تلویزیونی معمولی (آنالوگ) مورد استفاده قرار می‌گیرند.

اگر هر یک از ۵۰ کانال باقیمانده را با QAM-256 مدوله کنیم، پهنای باندی معادل 40 Mbps برای هر کانال (و در مجموع 2 Gbps) بدست می‌آوریم. با نزدیکتر کردن جعبه تقسیم سیستم به خانه‌های مشتریان، می‌توان به هر ۵۰۰ خانه یک کابل رساند. با یک حساب سرانگشتی می‌توان دید که به هر خانه پهنای باندی معادل 4 Mbps می‌رسد، که برای پخش فیلمهای MPEG-2 کافیست.

هیجان‌انگیز است، نه؟ اما این کار مستلزم آن است که شرکتهای تلویزیون کابلی تمام کابلهای قدیمی را با کابلهای 750 MHz جایگزین کنند، جعبه تقسیمهای جدید نصب کنند، و تمام تقویت‌کننده‌های یکطرفه را هم جمع کنند – و این یعنی عوض کردن کل سیستم تلویزیون کابلی. هزینهٔ این کار با ایجاد یک زیرساخت کامل FTTC قابل مقایسه است. در هر دو سیستم، شرکتهای عمل‌کننده مجبورند به هر کوچه و خیابانی فیبر نوری بکشند، و در انتهای هر رشته فیبر یک مبدل نوری-الکتریکی نصب کنند. تنها فرق آنها در اینست که، در FTTC به هر خانه یک زوج سیم تابیده مستقل می‌رود، ولی در HFC یک کابل کواکس مشترک بین تمام خانه‌ها کشیده می‌شود. همانطور که می‌بینید، این دو سیستم آنقدرها که صاحبان آنها ادعا می‌کنند، با هم متفاوت نیستند.

با این حال یک تفاوت اساسی هست که باید به آن اشاره کرد: HFC از یک رسانهٔ مشترک (بدون هیچگونه سوئیچینگ و مسیردهی) استفاده می‌کند. هر اطلاعاتی که روی این کابل فرستاده شود، بطور بالقوه بوسیلهٔ تمام مشترکان قابل دریافت است. اما FTTC سیستمی است مبتنی بر سوئیچینگ، و چنین چیزی در آن اتفاق نمی‌افتد. اگر متولیان HFC بخواهند کسی بدون پرداخت پول قادر به تماشای فیلمهای پخش شده نباشد، باید از نوعی رمزنگاری استفاده کنند. در FTTC هیچ نیازی به رمزنگاری و اقدامات امنیتی اضافی نیست، و تنها حاصل آن افزایش پیچیدگی و افت کارایی سیستم خواهد بود. از دیدگاه شرکتهای پخش رسانه، رمزنگاری ایدهٔ خوبیست یا خیر؟ اغلب شرکتهای تلفن چنین کاری نمی‌کنند، با این توضیح که مایل به افت کیفیت نیستند (ولی در واقع برای ضرر زدن به رقبای HFC).

بعد از این سیستمهای توزیع، نوبت به سرویس‌دهنده‌های ویدئوی محلی می‌رسد. آنها در واقع نسخهٔ کوچکی از سرویس‌دهنده‌های ویدئو (که در بالا توضیح دادیم) هستند. نقش اصلی این سرویس‌دهنده‌ها کاستن از ترافیک ستون فقرات است.

از این سرویس‌دهنده‌های محلی می‌توان برای رزرو کردن فیلمها استفاده کرد. اگر مشتریان از مدتی قبل تمایل خود را برای دیدن فیلمی به شرکت پخش رسانه اعلام کنند، این فیلم می‌تواند در ساعات خلوت شبکه در سرویس‌دهندهٔ محلی بار شود. این روش می‌تواند به کاهش قیمتها نیز منجر شود. برای مثال، اپراتور شبکه می‌تواند برای ساعات کم مصرف در تعرفه‌های خود تخفیف دهد.

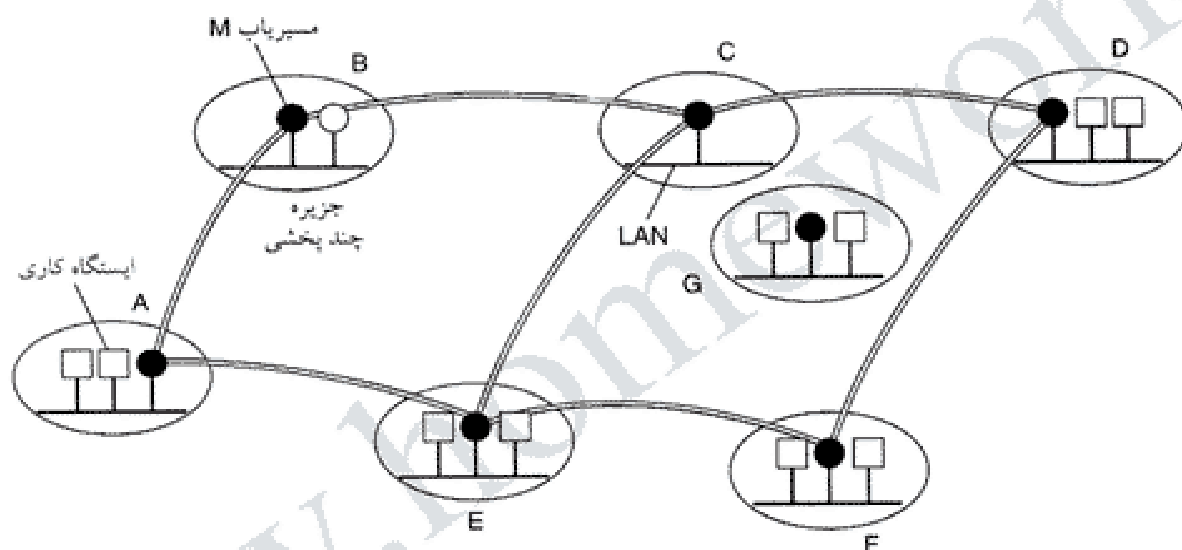
۷-۴-۹ ستون‌فقرات چندپخشی – MBone

در همان حالیکه شرکتهای تلفن و تلویزیون کابلی در حال طرح نقشه‌های بزرگ برای آیندهٔ «VOD دیجیتال» هستند، جامعهٔ اینترنت به آرامی در حال پیاده‌سازی سیستم چندرسانه‌ای دیجیتال خاص خود است: MBone (ستون فقرات چندپخشی – Multicast Backbone). در این قسمت خواهید دید که MBone چیست، و چگونه کار می‌کند.

MBone را می‌توان تلویزیون اینترنتی دانست. بر خلاف VOD (که تأکید روی تماس کاربر و پخش فیلمهای از پیش فشرده‌شده از روی یک سرویس‌دهندهٔ ویدئو است)، MBone بیشتر برای پخش ویدئوی دیجیتالی زنده

از سراسر دنیا و از طریق اینترنت مورد استفاده قرار می‌گیرد. این سیستم از سال ۱۹۹۲ عملیاتی شده، و کنفرانسهای علمی، از جمله گردهمایی‌های IETF ، و رخدادهای مهم علمی (مانند پرتاب شاتل فضایی) از طریق آن پخش شده است. یکی از کنسرتهای گروه رولینگ استونز، و بخشهایی از جشنوارهٔ فیلم کن نیز از طریق MBone پخش شده‌اند (البته در اینکه بتوان اینها را رخدادهای مهم علمی بشمار آورد، جای بحث است).

از نظر فنی، MBone شبکه‌ایست مجازی روی اینترنت، که تشکیل شده است از تعدادی **جزیرهٔ چندپخشی** (multicast island) که بوسیلهٔ چند **تونل** (tunnel) به هم متصل شده‌اند (شکل ۷–۸۱ را ببینید). در این شکل، MBone شش جزیره دارد (A تا F)، که بوسیلهٔ هفت تونل به هم وصل شده‌اند. هر جزیره (که یک LAN ، یا چند LAN متصل به هم است) از چندپخشی سخت‌افزاری به کامپیوترهای میزبان خود پشتیبانی می‌کند. بسته‌های MBone از طریق تونل‌ها بین جزایر منتشر می‌شود. شاید روزی در آینده، که تمام مسیرهای اینترنت توانایی جابجایی ترافیک چندپخشی را بدست آورند، دیگر نیازی به این ساختار نباشد، ولی فعلاً که کار ما را راه می‌اندازد.



شکل ۷–۸۱ MBone تشکیل شده است از جزایر چندپخشی، که بوسیلهٔ تونل به هم متصل شده‌اند.

هر جزیره دارای یک یا چند مسیریاب ویژه موسوم به **مسیریاب چندپخشی** (mrouter - multicast router) است. برخی از اینها مسیریابهای معمولی هستند، ولی برخی دیگر کامپیوترهای یونیکس بیش نیستند که نرم‌افزار خاصی را (در سطح root) اجرا می‌کنند. مسیریابهای چندپخشی از نظر منطقی به تونل‌ها متصل می‌شوند. بسته‌های MBone در داخل بسته‌های IP پیچیده شده و بصورت بسته‌های **تک‌پخشی** (unicast) معمولی به آدرس مسیریاب چندپخشی مقصد ارسال می‌شوند.

تونل‌ها را باید بصورت دستی پیکربندی کرد. معمولاً، تونل روی یک مسیر فیزیکی اجرا می‌شود، ولی این الزامی نیست. مثلاً، اگر مسیر فیزیکی یک تونل از بد حادثه دچار مشکل شود، مسیریابهای چندپخشی (که از این تونل استفاده می‌کنند) حتی از این موضوع مطلع نخواهند شد، چون اینترنت بطور خودکار ترافیک بسته‌های IP را به مسیرهای دیگر هدایت خواهد کرد.

وقتی یک جزیرهٔ جدید می‌خواهد به MBone وصل شود (مانند G در شکل ۷–۸۱)، سرپرست آن با ارسال پیام به لیست پستی MBone حضور خود را اعلام می‌کند. پس از آن سرپرستان سایتهای مجاور با وی تماس می‌گیرند، تا وی بتواند تونل‌های موردنیاز را پیکربندی کند. حتی گاهی اوقات تونل‌های موجود برای بهینه کردن

توپولوژی شبکه و بهره‌گیری از جزیرهٔ جدیدالتأسیس، آرایش خود را عوض می‌کنند. چون آنها که واقعاً وجـود خارجی ندارند، و چیزی نیستند جز چند جدول در حافظهٔ مسیریابها (که آنها را هم براحتی می‌توان اضافه، حذف، و یا جابجا کرد). معمولاً هر کشور متصل به MBone دارای یک ستون‌فقرات و تعدادی جزیرهٔ متصل به این ستون فقرات است. با عبور یک یا دو تونل از اقیانوس اطلس، MBone به شبکه‌ای جهانی تبدیل شده است.

بنابراین، MBone صرفنظر از تعداد آدرسهای چندپخشی (و تعداد کسانی که به آن گوش می‌کنند، یا آنرا تماشا می‌کنند) چیزی نیست جز چند جزیره و تونل، که کاملاً شبیه یک زیرشبکهٔ معمولی (فیزیکی) است، بنابراین می‌توان از الگوریتمهای معمولی مسیریابی برای آن استفاده کرد. به همین دلیل، MBone در شروع کار از یک الگوریتم مسیریابی بر اساس الگوریتم بردار فاصلهٔ بل‌من-فورد، بنام DVMRP (**پروتکل مسیریابی چندپخشی بردار فاصله** – Distance Vector Multicast Routing Protocol)، استفاده کرد. برای مثال، در شکل ۷–۸۱ جزیرهٔ *C* برای ارسال بسته‌های خود به جزیرهٔ *A* می‌تواند از طریق *B* یا *E* (و یا حتی *D*) اقدام کند. این جزیره برای انتخاب مسیر مناسب، فاصلهٔ هر گره تا *A* را (از خود آن گره) گرفته و آنها را با هم جمع می‌کند. با این روش، هر جزیره می‌تواند بهترین مسیر به جزیره‌های دیگر را محاسبه کند. اما (همانطور که بزودی خواهید دید) مسیرها واقعاً به این شکل استفاده نمی‌شوند.

ابتدا اجازه دهید ببینیم اصلاً چندپخشی چگونه اتفاق می‌افتد. برای این که یک کامپیوتر بتواند صدا یا فیلم پخش کند، ابتدا باید یک آدرس چندپخشی کلاس D بگیرد. آدرسهای کلاس D مانند فرکانس ایستگاه رادیویی عمل می‌کنند، و همهٔ آنها در یک پایگاه دادهٔ واحد نگهداری می‌شوند. یک کامپیوتر می‌تواند بـه هـر آدرس چندپخشی که مایل است گوش کند، درست مثل تنظیم موج رادیو.

مسیریابهای چندپخشی بصورت متناوب بسته‌های پخشی IGMP در جزیرهٔ خود منتشر می‌کنند، تا ببینند چه کسی به کدام کانالها علاقه دارد. میزبانهایی که بخواهند آن کانال را دریافت کنند، در پاسخ یک بستهٔ IGMP برمی‌گردانند. این پاسخها هم به تناوب فرستاده می‌شوند، تا شبکهٔ محلی در این بسته‌ها غرق نشود. هر مسیریاب چندپخشی جدولی دارد که نشان می‌دهد چه کانالهایی را باید روی LAN خود پخش کند (پخش کانالهایی که هیچ بیننده‌ای ندارند، چیزی جز اتلاف پهنای باند نیست).

برنامه‌های چندپخشی بطریق ذیل روی MBone منتشر می‌شوند. وقتی منبع صدا یا ویدئو بستهٔ جدیدی تولید می‌کند، آن بسته را (به کمک سخت‌افزار چندپخشی) در جزیرهٔ خود پخش می‌کند. مسیریاب چـندپخشی ایـن بسته‌ها را گرفته، و روی تمام تونلهایی که به آنها وصل است، پخش می‌کند.

هر مسیریاب چندپخشی که بسته‌ای دریافت می‌کند، ابتدا چک می‌کند که آیا این بسته بهترین مسیر را طی می‌کند (هر مسیریاب بهترین مسیرها برای تمام گره‌ها را در جدولی نگه می‌دارد). اگر بسته در بهترین مسیر خود به این مسیریاب رسیده باشد، مسیریاب آنرا روی تمام تونلهای خروجی (تمام تونلها منهای تونلی که بسته از آن وارد شده) کپی می‌کند. اگر بسته در بهترین مسیر نباشد، مسیریاب آنرا دور می‌اندازد. برای مثال، اگر در شکل ۷–۸۱ جدول مسیریاب *C* بگوید که بهترین مسیر به *A* از گره *B* می‌گذرد، و یک بستهٔ چندپخشی از *A* و از طریق *B* به *C* برسد، مسیریاب *C* این بسته را به تونلهای متصل به *D* و *E* کپی می‌کند. اما اگر یک بسته از *E* طریق از *A* به *C* رسیده باشد (که طبق جدول، بهترین مسیر نیست)، مسیریاب *C* آنرا دور می‌اندازد. اگر بخاطر داشته باشید، این همان الگوریتم هدایت در مسیر معکوس است (که در فصل ۵ دیدید)، و با اینکه کاملترین روش نیست ولی الگوریتمی ساده و نسبتاً خوب محسوب می‌شود.

برای جلوگیری از ازدحام در اینترنت، علاوه بر استفاده از الگوریتم هدایت در مسیر معکوس، بسته‌های IP به TTL (**زمان‌زنده‌ماندن** – Time To Live) نیز مجهز می‌شوند تا برای همیشه در اینترنت سرگردان نشوند. ایـن

مقدار توسط منبع ارسال بستهٔ چندپخشی تعیین می‌شود. هر تونل MBone دارای یک وزن خاص است، و اگر بسته‌ای بخواهد از یک تونل عبور کند، باید وزن کافی داشته باشد - در غیر اینصورت دور انداخته خواهد شد. برای مثال، اگر به تونل بین قاره‌ای (که از اقیانوس اطلس عبور می‌کند) وزن 128 بدهیم، و بخواهیم یک بستهٔ چندپخشی را در همان قارهٔ مبدأ محبوس کنیم و نگذاریم به قارهٔ دیگر برود، کافیست فیلد TTL آنرا به 127 (یا کمتر) ست کنیم. وقتی یک بسته از تونلی عبور می‌کند، باندازهٔ وزن تونل از TTL آن کم خواهد شد.

با اینکه الگوریتم مسیریابی فوق بخوبی کار می‌کند، محققان زیادی برای بهبود آن تلاش می‌کنند. در یکی از این پیشنهادات از ایدهٔ مسیریابی بردار فاصله، ضمن تقسیم سلسله مراتبی سایتهای MBone به مناطق مختلف، بهره گرفته شده است (Thyagarajan and Deering, 1995) . پیشنهاد دیگر استفاده از مسیریابی حالت لینک، بجای مسیریابی بردار فاصله، است. بویژه، یکی از گروههای کاری IETF تغییراتی در پروتکل OSPF داده تا برای ایجاد یک **سیستم چندپخشی خودمختار** (Multicast AS) مناسب شود. این پروتکل **OSPF چندپخشی** (یا MOSPF) نام دارد (Moy, 1994) . در MOSPF ، مسیریاب علاوه بر اطلاعات معمولی مسیریابی، نقشهٔ کامل جزیره‌ها و تونلهای چندپخشی را هم نگه می‌دارد. با استفاده از این اطلاعات، پیدا کردن بهترین مسیر از هر جزیره به جزیرهٔ دیگر کاری ساده و سر راست خواهد بود. الگوریتم دایکسترا (Dijkstra) یکی از الگوریتمهایی است که می‌توان از آن استفاده کرد.

زمینهٔ دیگر تحقیقات مسیریابی بین-AS (بین سیستمهای خودمختار) است. در اینجا نیز یکی از گروههای کاری IETF الگوریتمی بنام PIM (**چندپخشی مستقل از پروتکل** - Protocol Independent Muliticast) توسعه داده است. این پروتکل دو ویرایش دارد: یکی برای مناطق شلوغ (جزیره‌های پُربیننده)، بنام PIM-DM ؛ و دیگری برای مناطق خلوت (جزیره‌های کم‌بیننده)، بنام PIM-SM . هر دو ویرایش، بجای ایجاد لایه‌های توپولوژی خاص مثل کاری که DVMRP و MOSPF انجام می‌دهند، از جدولهای مسیریابی تک‌پخشی استاندارد استفاده می‌کنند.

در PIM-DM مسیرهای زائد و بی‌مصرف حذف می‌شوند. روش حذف شاخه‌های اضافی چنین است. وقتی یک بستهٔ چندپخشی از تونل «اشتباه» به یک گره می‌رسد، این گره از طریق همان تونل یک **بستهٔ حذف** (purne packet) به فرستنده برمی‌گرداند، و از وی می‌خواهد که تا دیگر بسته‌ای در آن مسیر به وی ندهد. وقتی همین گره یک بستهٔ «صحیح» دریافت می‌کند، آنرا به تمام تونلهایی که قبلاً خودشان را (برای آن مسیر) حذف نکرده‌اند، کپی می‌کند. اگر تمام تونلها خود را حذف کرده باشند، و در همان جزیره هم شنونده یا بیننده‌ای برای آن بسته وجود ندارد، مسیریاب چندپخشی یک بستهٔ حذف روی تونل «صحیح» برمی‌گرداند. بدین‌ترتیب، چندپخشی همیشه بطور خودکار خود را با بهترین مسیرها منطبق می‌کند.

طرز کار PIM-SM ، که در RFC 2362 تعریف شده، متفاوت است. در اینجا هدف آن است که برای یک کنفرانس سه نفره روی یک آدرس کلاس D در برکلی آمریکا، تمام مسیرهای اینترنت مشغول نشود. در این رهیافت از مفهومی بنام **نقطهٔ قرار** (rendezvous point) استفاده می‌شود. هر فرستنده در یک گروه چندپخشی PIM-SM بسته‌های خود را به یک نقطهٔ قرار می‌فرستد، و هر سایتی که به محتویات این فرستنده علاقمند باشد، یک تونل به آن نقطهٔ قرار ایجاد می‌کند. با این روش، ترافیک PIM-SM از حالت چندپخشی خارج شده و بصورت تک‌پخشی درمی‌آید. محبوبیت PIM-SM روز به روز در حال افزایش است، و MBone بتدریج به آن سمت حرکت می‌کند (و MOSPF کم‌کم کاربرد خود را از دست می‌دهد). البته خود MBone هم بتدریج به نقطهٔ اشباع و رکود نزدیک می‌شود، و بنظر می‌رسد هرگز نتواند به موفقیت خیره‌کننده‌ای دست یابد.

حتی اگر MBone موفقیت آنچنانی بدست نیاورد، شبکه‌های چندرسانه‌ای همچنان یکی از فیلدهای مهیج و رشدیابندهٔ اینترنت باقی خواهند ماند. هر روز تکنولوژی‌ها و برنامه‌های جدیدی وارد بازار می‌شوند، و چندپخشی

و کیفیت سرویس در حال نزدیک شدن به یکدیگر هستند (Striegel and Manimaran, 2002) . چندپخشی بیسیم یکی دیگر از زمینه‌های پُرتوجه و داغ است (Gossain et al., 2002) . چندپخشی و تمام زمینه‌های مرتبط با آن احتمالاً تا سالهای آینده در مرکز توجه باقی خواهند ماند.

۷ـ۵ خلاصه

برای نامگذاری در اینترنت از یک سیستم سلسله‌مراتبی بنام سیستم نام ناحیه (DNS) استفاده می‌شود. در بالاترین نقطهٔ این هرم ناحیه‌های شناخته شده، مانند *com* ، *edu* و نزدیک به ۲۰۰ ناحیهٔ کشوری، قرار دارد. DNS یک پایگاه‌دادهٔ توزیع شده است، که سرویس دهنده‌های آن در تمام دنیا پخش هستند. وظیفهٔ DNS تبدیل نام ناحیه به آدرس IP است.

یکی از پُرطرفدارترین کاربردهای اینترنت ایمیل (email) است، و امروزه همه، از بچه‌های مدرسه‌ای گرفته تا پدربزرگها و مادربزرگها، از آن استفاده می‌کنند. اغلب سیستمهای ایمیل امروزی با استانداردهای تعریف شده در RFC 2821 و RFC 2822 کار می‌کنند. در این سیستمها مشخصات پیام به کمک سرآیندهای متنی (ASCII) تعیین می‌شود، و برای ارسال محتویات پیام می‌توان از انواع دادهٔ MIME استفاده کرد. ارسال پیام با برقراری یک اتصال TCP به پورت 25 کامپیوتر مقصد، به کمک پروتکلی بنام SMTP ، انجام می‌شود.

شبکهٔ تارنمای جهانی (وب) نیز تقریباً به همان اندازهٔ ایمیل طرفدار دارد. وب سیستمی است از سندهای لینک شده به یکدیگر؛ این سندها به زبان HTML نوشته می‌شوند. در وب محتویات دینامیک نیز، در هر دو نوع سمت-سرویس‌دهنده (PHP ، JSP ، و ASP) و سمت-مشتری (JavaScript ، و VBScript)، وجود دارند. برای دیدن صفحات وب از برنامه‌هایی موسوم به مرورگر، که کار آنها اتصال به سرویس‌دهنده، دریافت صفحهٔ وب و نمایش آن است، استفاده می‌شود. برای افزایش کارایی وب تکنیکهای مختلفی، مانند حافظهٔ نهان، تکثیر سرویس‌دهنده و شبکه‌های تحویل محتوا، توسعه داده شده است.

وب بیسیم تازه در آغاز راه است. اولین آنها عبارتند از WAP و I-Mode ، که پهنای باند کمی دارند و صفحهٔ نمایش آنها نیز کوچک است؛ ولی نسل بعدی قویتر خواهد بود.

چندرسانه‌ای (صدا و تصویر دیجیتال) یکی دیگر از ستاره‌های در حال طلوع اینترنت است. صدای دیجیتال (صدای جویباری، صدا روی IP ، و رادیوی اینترنتی) بدلیل نیاز به پهنای باند کمتر مدتهاست وارد بازار شده، و همچنان در حال رشد است. پخش فیلم بر حسب تقاضا نیز طرفداران زیادی دارد، و در آینده از آن بیشتر خواهید شنید. یکی دیگر از بازیگران این صحنه MBone (سرویس تلویزیون دیجیتال اینترنتی) هنوز در مراحل تجربی بسر می‌برد.

مسائل

۱. اغلب کامپیوترهای تجاری سه شناسهٔ جهانی و منحصربفرد دارند. آنها چیستند؟

۲. با توجه به اطلاعات داده شده در شکل ۷-۳، کامپیوتر *little-sister.cs.vu.nl* جزء کلاس A است، یا B یا C ؟

۳. در شکل ۷-۳ بعد از *rowboat* نقطه وجود ندارد. چرا؟

۴. حدس بزنید خندانک :-X (که گاهی به شکل :-# هم نوشته می‌شود) چه معنایی دارد.

۵. DNS بجای TCP از UDP استفاده می‌کند، و اگر بسته‌ای گم شود، بازیابی آن بطور خودکار ممکن نیست. آیا این مشکل‌ساز نیست؟ اگر هست، راه حل آن چیست؟

۶. بسته‌های UDP ، علاوه بر احتمال گم شدن، محدودیت طول نیز دارند (حداکثر ۵۷۶ بایت). اگر یک نام

DNS بلندتر از این باشد، چه باید کرد؟ آیا می‌توان آنرا در دو بسته فرستاد؟

۷. آیا ماشینی با یک نام DNS می‌تواند چند آدرس IP داشته باشد؟ چگونه؟

۸. آیا یک کامپیوتر می‌تواند دو نام DNS (در دو ناحیهٔ کاملاً جدا) داشته باشد؟ اگر جواب مثبت است، یک مثال بزنید. اگر نه، چرا؟

۹. در سالهای اخیر تعداد شرکتهایی که سایت وب دارند، بصورت انفجاری افزایش یافته، و اغلب این شرکتها هم سایت خود را در ناحیهٔ *com* ثبت کرده‌اند، که باعث فشار بسیار زیاد به سرویس‌دهنده‌های سطح-بالا در این ناحیه شده است. آیا برای رفع این مشکل (بدون تغییر دادن ساختار سیستم و معرفی ناحیهٔ جدید) راهی می‌شناسید؟ اگر این راه حل به تغییر در سمت مشتری متکی باشد، اشکالی ندارد.

۱۰. در برخی از سیستمهای ایمیل فیلدی در سرآیند پیامها وجود دارد بنام *Content Return:* - این فیلد مشخص می‌کند که اگر گیرنده در مقصد شناسایی نشد، بدنهٔ پیام برگشت داده شود یا خیر. این فیلد در کدام قسمت قرار می‌گیرد: پاکت نامه، یا سرآیند آن؟

۱۱. سیستمهای پُست‌الکترونیک به نوعی دایرکتوری نیاز دارند، تا بتوانند اشخاص را به کمک آن پیدا کنند. برای ایجاد چنین دایرکتوری، مشخصات افراد باید به قسمتهای کوچکتر (نام، نام خانوادگی، و مانند آن) شکسته شود. دربارهٔ مشکلات تدوین استاندارد بین‌المللی چنین دایرکتوری بحث کنید.

۱۲. آدرس ایمیل افراد عبارتست از: نام ورود فرد به سیستم @ نام ناحیهٔ DNS با یک رکورد *MX* . این نام ورود می‌تواند هر چیزی (نام کوچک، نام خانوادگی، و یا هر ترکیبی از آن با حروف و اعداد دیگر) باشد. در یک شرکت تعداد زیادی از ایمیل‌ها به هدر می‌رود، چون افراد نام ورود به سیستم گیرنده‌ها را نمی‌دانند. آیا راهی برای حل این مشکل بدون تغییر دادن DNS وجود دارد؟ اگر بله، چگونه؟ اگر خیر، چرا؟

۱۳. اندازهٔ یک فایل باینری ۳۰۷۲ بایتی بعد از تبدیل به کُد base64 (با یک جفت CR+LF بعد از هر ۸۰ بایت) چقدر خواهد شد؟

۱٤. روش کُدگذاری quoted-printable را در نظر بگیرید. یکی از مشکلات این روش را که در متن کتاب به آن اشاره نشده، نام برده و دربارهٔ راه حل آن بحث کنید.

۱۵. پنج نوع دادهٔ MIME را که در کتاب نیامده، نام ببرید. برای این کار می‌توانید در اینترنت جستجو کنید.

۱۶. فرض کنید می‌خواهید یک فایل MP3 را از طریق ایمیل به دوست خود بفرستید، ولی ISP دوست شما روی ایمیلهای وارده محدودیت حجم 1 MB اعمال کرده، در حالیکه فایل شما 4 MB است. آیا راهی برای غلبه بر این مشکل (با توجه به RFC 822 و MIME) وجود دارد؟

۱۷. فرض کنید فردی یک دیمون تعطیلات برای خود ایجاد کرده، و درست قبل از خروج یک ایمیل به دوست خود می‌فرستد. متأسفانه این دوست هم برای یک هفته به مرخصی رفته و یک دیمون تعطیلات برای خود ست کرده است. چه اتفاقی می‌افتد؟ آیا این ایمیل‌ها تا برگشتن یکی از این دو نفر مدام بین دو سیستم پاس کاری خواهد شد؟

۱۸. در هر استانداردی، مانند RFC 822 ، قواعد گرامری (حتی ساده‌ترین آنها) باید به دقت تعریف شوند تا سیستمهای مختلف بتوانند با یکدیگر کار کنند. در سرآیندهای SMTP می‌توان از فاصلهٔ سفید (white space - هر یک از کاراکترهای فاصله، *Enter* یا *Tab*) بین توکن‌ها استفاده کرد. دو جایگزین ممکن دیگر برای فاصلهٔ بین توکن‌ها چیست؟

۱۹. دیمون تعطیلات بخشی از عامل کاربر است، یا عامل انتقال پیام؟ البته برای ست کردن دیمون تعطیلات از

عامل کاربر استفاده می‌کنیم، ولی آیا پاسخها را همین عامل برمی‌گرداند؟ توضیح دهید.

۲۰. پروتکل POP3 اجازه می‌دهد تا کاربران ایمیلهای خود را از یک صندوق پستی راه دور دریافت کنند. آیا این بدان معناست که فرمت داخلی صندوقهای POP3 باید استاندارد باشد، تا سیستمهای مختلف بتوانند با آن کار کنند؟ توضیح دهید.

۲۱. پروتکلهای POP3 و IMAP از دیدگاه یک ISP تفاوت فاحشی دارند: کاربران POP3 معمولاً صندوق پستی خود را خالی می‌کنند، در حالیکه کاربران IMAP نامه‌های خود را برای مدتی نامحدود روی سرویس‌دهنده نگه می‌دارند. شما کدام روش را به یک ISP پیشنهاد می‌کنید؟ دلایل خود را توضیح دهید.

۲۲. پُست وب (Webmail) از POP3 استفاده می‌کند، یا IMAP ، یا هیچکدام؟ چرا؟ اگر پاسخ منفی است، روش آن به کدامیک نزدیکتر است؟

۲۳. هنگام ارسال صفحات وب هم از سرآیندهای MIME استفاده می‌شود. چرا؟

۲٤. چه زمانی از برنامه‌های کمکی برای دیدن محتویات صفحهٔ وب استفاده می‌شود؟ مرورگر چگونه تشخیص می‌دهد از کدام برنامه باید استفاده کند؟

۲۵. آیا امکان دارد کلیک کردن روی یک لینک در مرورگرهای نت‌اسکیپ و اینترنت اکسپلورر باعث اجرای برنامه‌های کمکی متفاوتی شود (در حالیکه در آن لینک یک نوع MIME مشخص شده است)؟ توضیح دهید.

۲۶. در شکل ۷-۲۱ یک سرویس‌دهندهٔ وب چندریسمانی را ملاحظه می‌کنید. در این سرویس‌دهنده گرفتن درخواست کاربر و چک کردن حافظهٔ نهان 500 μsec طول می‌کشد. در نیمی از مواقع فایل درخواستی در حافظهٔ نهان پیدا شده، و بلافاصله برگردانده می‌شود. در نیمی دیگر، این ماژول باید برای خواندن دیسک 9 msec دیگر صبر کند. (با فرض اینکه دیسک گلوگاه سیستم نباشد) برای استفادهٔ کامل از ظرفیت CPU سرویس‌دهنده چند ماژول باید اجرا کند؟

۲۷. در URL های استاندارد *http* فرض بر این است که سرویس‌دهنده به پورت 80 گوش می‌کند، ولی این بهیچوجه اجباری نیست. روشی طراحی کنید که URL بتواند به پورتهای غیراستاندارد هم دسترسی پیدا کند.

۲۸. URL ها می‌توانند علاوه بر نامهای DNS مستقیماً از آدرس IP هم استفاده کنند (مانند، *http://192.31.231.66/index.html*). مرورگر چگونه می‌تواند تشخیص دهد که بعد از *http://* یک نام DNS آمده یا آدرس IP ؟

۲۹. فرض کنید فردی در دپارتمان کامپیوتر دانشگاه استنفورد برنامه‌ای نوشته و می‌خواهد آنرا از طریق FTP توزیع کند. محل این فایل در کامپیوتر سرویس‌دهنده *ftp/pub/freebits/newprog.c* است - URL آن چه می‌تواند باشد؟

۳۰. در شکل ۷-۲۵، سایت *www.portal.com* تنظیمات کاربر را در یک کوکی نگه می‌دارد. مشکل این روش آن است که کوکی به 4 KB محدود است، و ممکنست برای نگهداری تنظیمات کاربر کافی نباشد. روشی طراحی کنید که این محدودیت را نداشته باشد.

۳۱. یک بانک تجاری («بانک تنبل‌ها») که می‌خواهد مشتریان تنبل خود را هم راضی نگه دارد، سیستمی طراحی کرده که بعد از وارد شدن مشتری به سیستم (با نام کاربر و کلمهٔ رمز) یک شمارهٔ شناسایی بصورت کوکی روی کامپیوتر وی ذخیره می‌کند تا کاربر مجبور نباشد هر بار نام کاربر و کلمهٔ رمز خود را وارد کند. چه نظری

دربارهٔ این ایده دارید؟ آیا عملی است؟ آیا ایدهٔ خوبیست؟

۳۲. در شکل ۷-۲۶، پارامتر *ALT* برچسب <img> ست شده است. تحت چه شرایطی مرورگر از این پارامتر استفاده می‌کند (و چگونه)؟

۳۳. چگونه می‌توان یک تصویر را در HTML قابل‌کلیک کرد؟ مثالی بزنید.

۳٤. با استفاده از برچسب <a> برای کلمهٔ ACM لینکی به آدرس http://www.acm.org تعریف کنید.

۳۵. یک فرم سفارش خرید برای شرکت «اینترنت همبرگر» طراحی کنید، که مشتریان آن بتوانند از طریق اینترنت همبرگر خریداری کنند. در این فرم مشتری مشخصات خود (نام، آدرس، و شهر محل سکونت) و البته اندازهٔ همبرگر (بزرگ یا بسیار بزرگ) و نوع پنیر آنرا وارد می‌کند. پول همبرگرها هنگام تحویل نقداً دریافت می‌شود، و امکان خرید با کارت اعتباری وجود ندارد.

۳۶. فرمی طراحی کنید که کاربر دو عدد را وارد کرده، و وقتی دکمهٔ Submit را کلیک کرد، سرویس‌دهنده جمع آنها رابرگرداند. برای نوشتن اسکریپت سمت–سرویس‌دهنده از PHP استفاده کنید.

۳۷. برای هر یک از برنامه‌های زیر، مشخص کنید که آیا امکان استفاده از PHP یا جاوااسکریپت وجود دارد. و کدامیک بهتر است.

(الف) نمایش یک تقویم برای هر یک از ماههای بعد از سپتامبر ۱۷۵۲.

(ب) نمایش جدول پروازهای آمستردام به نیویورک.

(ج) رسم نمودار چندجمله‌ای با استفاده از ضرایبی که کاربر وارد می‌کند.

۳۸. برنامه‌ای با جاوااسکریپت بنویسید، که یک عدد بزرگتر از ۲ را گرفته و مشخص کند این عدد اول است یا خیر. توجه کنید که جاوااسکریپت دارای ساختارهای if و while (شبیه C و جاوا) است. عملگر باقیمانده در جاوااسکریپت % است. اگر به جذر x نیاز داشتید، می‌توانید از تابع *Math.sqrt(x)* استفاده کنید.

۳۹. صفحهٔ HTML زیر را در نظر بگیرید:

```
<html> <body>
<a href="www.info-source.com/welcome.html">Click here for info</a>
</body> </html>
```

وقتی کاربر این لینک را کلیک می‌کند، یک اتصال TCP به سرویس‌دهنده برقرار شده و چند خط به آن فرستاده می‌شود. این خط‌ها را بنویسید.

٤۰. از سرآیند *If-Modified-Since* می‌توان برای تعیین اعتبار صفحهٔ وب استفاده کرد. هر درخواست می‌تواند شامل مختلف (تصویر، صدا، ویدئو و البته HTML) باشد. کارایی این تکنیک را در مورد تصاویر JPEG و HTML مقایسه کنید.

٤۱. در روزهایی که مسابقات مهم ورزشی برگزار می‌شود، تعداد مراجعات به سایت رسمی آن رویداد افزایش چشمگیری پیدا می‌کند. آیا این اتفاق شبیه انتخابات فلوریدا در سال ۲۰۰۰ است؟ توضیح دهید.

٤۲. آیا یک ISP منفرد می‌تواند بعنوان CDN عمل کند؟ اگر بله، چگونه؟ اگر خیر، چرا؟

٤۳. در چه شرایطی استفاده از CDN ایدهٔ مناسبی نیست؟

٤٤. ترمینالهای بیسیم وب پهنای باند کمی دارند، و بهمین دلیل کُد کردن مناسب اهمیت بیشتری می‌یابد. روشی با کارایی مناسب برای انتقال متن انگلیسی روی لینکهای بیسیم به دستگاههای WAP طراحی کنید. فرض کنید دستگاه WAP چندین مگابایت ROM و CPU نسبتاً قوی دارد. راهنمایی: از ایدهٔ زبان ژاپنی، که در آن هر علامت معادل یک کلمه است، کمک بگیرید.

۴۵. یک دیسک فشردهٔ صوتی ظرفیتی معادل 650 MB دارد. آیا در این دیسکها از فشرده‌سازی استفاده می‌شود؟ توضیح دهید.

۴۶. در شکل ۷–۵۷(ج)، بدلیل استفاده از نمونه‌های ۴ بیتی برای نمایش ۹ سیگنال نویز کوانتیزه کردن رُخ می‌دهد. اولین نمونه، در 0 ، دقیق است ولی چند نمونهٔ بعدی دقیق نیستند. درصد خطا در نقاط 1/32 ، 2/32 و 3/32 چقدر است؟

۴۷. آیا از مدل روان‌شنوایی می‌توان برای کاستن از پهنای باند موردنیاز تلفن اینترنتی استفاده کرد؟ اگر بله، تحت چه شرایطی؟ اگر خیر، چرا؟

۴۸. فاصلهٔ یک سرویس‌دهندهٔ صدای جویباری با پخش‌کننده معادل 50 msec ، و سرعت ارسال آن 1 Mbps است. اگر پخش‌کننده دارای بافری 1 MB باشد، دربارهٔ علامتهای حد–بالا و پائین چه می‌توان گفت؟

۴۹. مزیت الگوریتم یک‌درمیانی (شکل ۷–۶۰) آن است که در صورت گم شدن چند بسته خللی در پخش پیش نمی‌آید. اما کاربرد این الگوریتم در تلفن اینترنتی دارای یک عیب کوچک نیز هست. آن چیست؟

۵۰. آیا VOIP نیز مانند صدای جویباری با دیوار آتش (فایروال) مشکل دارد؟ توضیح دهید.

۵۱. نرخ انتقال تصویر غیرفشردهٔ 600 × 800 با رنگ 8 bit و با سرعت 40 frames/sec چقدر است؟

۵۲. آیا ۱ بیت خطا در یک فریم MPEG می‌تواند روی چند فریم تأثیر بگذارد؟ توضیح دهید.

۵۳. یک سرویس‌دهندهٔ ویدئو با ۱۰۰,۰۰۰ مشتری را در نظر بگیرید، که هر مشتری در ماه دو فیلم تماشا می‌کند. نیمی از این فیلمها باید در ساعت ۸ بعد از ظهر پخش شود. این سرویس‌دهنده در هر لحظه (در همان ساعت) چند فیلم باید پخش کند؟ اگر هر فیلم به 4 Mbps پهنای باند نیاز داشته باشد، این سرویس‌دهنده به چند خط OC-12 نیاز دارد؟

۵۴. فرض کنید قانون **زیف** در مورد یک سرویس‌دهندهٔ ویدئو با ۱۰,۰۰۰ فیلم مصداق دارد. اگر این سرویس‌دهنده ۱۰۰۰ تا از فیلمهای پُربیننده‌تر را روی دیسک و ۹۰۰۰ فیلم دیگر را روی CD نگه دارد، چند درصد از درخواستها به دیسک مراجعه می‌کند؟ (عبارت آنرا بدست آورید.) برای ارزیابی عددی این فرمول، یک برنامهٔ کوچک بنویسید.

۵۵. برخی از متقلبان اینترنتی اقدام به ثبت نامهای اینترنتی نزدیک به نامهای معروف می‌کنند (مانند *www.microsfot.com* که فقط یک حرف آن با *www.microsoft.com* جابجا شده است). حداقل پنج تا از این نامهای تقلبی را فهرست کنید.

۵۶. بسیاری از افراد نامهای DNS بصورت *www.word.com* که در آن *word* یکی از کلمات رایج است، را ثبت کرده‌اند. رای هر یک از دسته‌های زیر پنج سایت (بهمراه مختصری از مشخصات آنها) فهرست کنید: حیوانات، غذاها، لوازم خانگی، و اندامهای بدن.

۵۷. چن[illegible] علام[illegible] دلخواه اموجی 12 × 12 طراحی کنید. برای مثال، سعی کنید کلمات پسر، دختر، معلم، و سیاستمدار را نمایش دهید.

۵۸. یک سرویس‌دهندهٔ POP3 بنویسید که فرمانهای زیر را قبول کند: *USER* ، *PASS* ، *LIST* ، *RETR* ، *DELE* ، و *QUIT* .

۵۹. سرویس‌دهندهٔ شکل ۶–۶ را با استفاده از فرمان HTTP 1.1 GET به یک سرویس‌دهندهٔ واقعی وب تبدیل کنید. این سرویس‌دهنده همچنین باید پیامهای *Host* را قبول کند. این سرویس‌دهنده باید یک حافظهٔ نه[illegible] داشته باشد، و فایلهایی را که اخیراً بازدید شده‌اند در این حافظهٔ نهان نگه دارد.



امنیت شبکه

در چند دههٔ ابتدایی پیدایش، از شبکه‌های کامپیوتری بیشتر توسط پژوهشگران دانشگاه و برای ارسال نامه‌های الکترونیکی و یا توسط کارمندان شرکتها برای به اشتراک‌گذاری چاپگر، استفاده می‌شد. در چنین شرایطی، امنیت شبکه از اهمیت چندانی برخوردار نبود. اما اکنون که میلیونها تن از شهروندان عادی از شبکه‌ها برای انجام عملیات بانکی، معاملات یا پر کردن اظهارنامه‌های مالیاتی خود استفاده می‌کنند، امنیت شبکه به عنوان یک مسئلهٔ بالقوه و عمده پدیدار شده است. در این فصل از چندین زاویه به مطالعهٔ امنیت شبکه خواهیم پرداخت، اشکالات و موانع امنیت را متذکر شده و چندین الگوریتم و پروتکل را که شبکه‌ها را امن‌تر و قابل‌اطمینان می‌کنند، تشریح خواهیم نمود.

«امنیت شبکه» در برگیرندهٔ عناوین و موارد بسیار گسترده است و بامشکلات و معضلات متعددی سروکار دارد. در یک عبارت ساده می‌توان امنیت را «اطمینان از عدم دسترسی افراد فضول و جلوگیری از دستکاری در پیامهای محرمانه دیگران» تعبیر کرد. همچنین می‌توان امنیت را در ارتباط با افرادی تعبیر کرد که تلاش می‌کنند به سرویسهای راه دور در شبکه دسترسی پیدا کنند در حالی که مجوز استفاده از آنها را ندارند؛ یا به روشهایی اطلاق می‌شود که بتوان صحت پیامهایی که مثلاً از ادارهٔ اخذ مالیات (IRS) می‌رسد و اعلام می‌کند: «حداکثر تا جمعه مبلغ اعلام شده را واریز کنید» را تائید کرد و تشخیص داد که این پیام واقعاً از ادارهٔ مالیات آمده نه از مافیا! همچنین می‌توان امنیت را در خصوص پیشگیری از دخل و تصرف و یا پاسخ جعلی به پیامهای قانونی دیگران و مقابله با افرادی که پس از ارسال پیام سعی در انکار آنها می‌کنند، تعبیر کرد.

منشاء اغلب مشکلاتی که به صورت عمدی برای امنیت شبکه‌ها به وجود می‌آید افرادی هستند که سعی در کسب درآمد نامشروع، جلب توجه یا آزاررسانی به دیگران دارند. در شکل ۸-۱ فهرستی از افراد که بطور عام مرتکب جرمهای امنیتی می‌شوند و انگیزه‌های آنان درج شده است. با بررسی این جدول روشن خواهد شد که تضمین امنیت شبکه، مقوله‌ای فراتر از رفع اشکالات برنامه‌نویسی است. در این خصوص باید تمهیداتی برای پیشگیری از حملهٔ دشمنانی اندیشیده شود که غالباً افرادی باهوش و کوشا هستند و گاهی اوقات سازمان‌یافته و با برنامه‌ریزی قبلی اقدام به حمله می‌کنند. البته همیشه عملیات مخرب بر علیه شبکه توسط یک گروه خاص و خطرناک انجام نمی‌شود؛ بررسی پرونده‌های ادارهٔ پلیس نشان می‌دهد که بسیاری از حملات بر علیه شبکه توسط عوامل خارجی و به واسطهٔ نفوذ از طریق خط تلفن نبوده است بلکه توسط عوامل مغرض داخلی انجام گرفته است. بنابراین طراحی سیستمهای امنیتی باید با در نظر داشتن این حقیقت انجام شود.

تهدیدکنندگان امنیت	هدف
دانشجو	تفریح کردن از طریق تجسس در نامه‌های دیگران
کراکر (Cracker)	به منظور آزمایش سیستم امنیت متعلق به شخص (یا گروه) خاص یا سرقت اطلاعات
نمایندهٔ فروش	برای اطلاع از طرحهای استراتژیک حریف در خصوص بازار خرید و فروش
کارمند اخراجی	برای انتقام‌گیری
حسابداران	برای اختلاس پول از یک شرکت
دلال سهام	برای انکار وعده‌هایی که به یک مشتری داده شده است (از طریق email)
کلاه‌بردار	برای سرقت شماره‌های کارتهای اعتباری جهت خرید
جاسوس	برای اطلاع از اسرار نظامی یا صنعتی دشمن
تروریستها	برای سرقت اسرار جنگهای میکروبی

شکل ۸-۱. فهرست برخی از افراد که منجر به مشکلات امنیتی می‌شوند و انگیزه‌های آنها.

مشکلات امنیت شبکه بطور کلی به چهار ردهٔ نزدیک و مرتبط به هم تقسیم‌بندی می‌شوند: (۱) **سرّی ماندن اطلاعات**[۱]، (۲) **احراز هویت کاربران**[۲]، (۳) **غیرقابل انکار بودن پیامها**[۳]، (۴) **نظارت بر صحّت اطلاعات**[۴].

سری ماندن اطلاعات که گاه «محرمانه نگاهداری اطلاعات» (Confidentiality) نیز نامیده می‌شود، متضمن انجام عملیاتی است که اطلاعات را از دسترس کاربران غیرمجاز و بیگانه دور نگاه می‌دارد. این همان مفهومی است که در ذهن مردم عادی در خصوص امنیت شبکه تداعی می‌شود.

«احراز هویت» عبارتست از تایید هویت طرف مقابل ارتباط قبل از آنکه اطلاعات حساس در اختیار او قرار بگیرد یا در معاملات تجاری شرکت داده شود.

مقولهٔ «غیرقابل انکار بودن پیامها» (Nonrepudiation) با امضاهای دیجیتالی سر و کار دارد و به اطلاعات و مستندات، هویت حقوقی اعطاء می‌کند: وقتی مشتری شما که قبلاً به صورت الکترونیک یک میلیون عدد از یک کالای کوچک به قیمت ۸۹ سنت سفارش داده و بعداً ادعا می‌کند قیمت کالا ۶۹ سنت بوده، چگونه می‌توان خلاف ادعای او را ثابت کرد؟ شاید حتی او ادعا کند هرگز چنین سفارش خریدی نداده است.

نهایتاً چگونه می‌توان مطمئن شد که پیامی که شما دریافت کرده‌اید، دقیقاً همان پیامی است که در اصل فرستاده شده و یک دشمن بدخواه در حین انتقال پیام آن را دستکاری و تحریف نکرده است.

تمام موارد فوق‌الذکر (سری ماندن اطلاعات، احراز هویت، غیرقابل انکار بودن پیامها و نظارت بر صحت اطلاعات) در سیستمهای سنتی و معمولی پیرامونمان نیز وجود دارد، البته با تفاوتهای قابل توجه؛ عملیات محرمانه نگاهداشتن و نظارت بر صحت اطلاعات با اتکاء به پست سفارشی و لاک و مهر کردن مستندات انجام می‌شود.

همچنین عموم افراد اغلب قادرند تفاوت بین اصل یک سند و تصویر آن سند را تشخیص بدهند. به عنوان یک آزمایش تصویر (کپی شده) یک چک معتبر بانکی را تهیه کرده و سعی در نقد کردن اصل چک در روز دوشنبه بنمائید. حال، تلاش کنید تصویر کپی شدهٔ چک را در روز سه‌شنبه نقد کنید تا تفاوت بین رفتاری که با شما می‌شود را عیناً مشاهده نمائید!!! در بانکداری الکترونیک، اصل و کپی چکهای الکترونیکی تفاوتی با هم نداشته و غیرقابل تشخیص است. لذا چگونگی برخورد بانک با اینگونه چکها جای تأمل دارد.

---

۱. Secracy ۲. Authentication ۳. Nonrepudiation ۴. Integrity Control

در دنیای پیرامون ما، افراد از طریق تشخیص چهرهٔ یک فرد، صدا یا دستخط، او را احراز هویت می‌کنند. در عملیات اداری، تأئید هویت اشخاص از طریق امضای آنها در برگه‌های حقوقی یا مهرهای برجسته و نظائر آن انجام می‌شود. هرگونه تلاش برای جعل یا دستکاری در اسناد، با مقایسهٔ دستخط یا امضاء، قابل کشف است. این گزینه‌ها در دنیای الکترونیک قابل اعمال نیستند و بدیهی است که به راهکارهای دیگری نیاز است.

قبل از آن که به ماهیت تک‌تک راه‌حلهای تضمین امنیت اطلاعات بپردازیم، بررسی تمهیدات و راهکارهای امنیتی که در هر یک از لایه‌های پشتهٔ پروتکلی شبکه (Protocol Stack) ممکن و قابل إعمال است، خالی از لطف نخواهد بود.

رعایت نکات و تمهیدات امنیتی فقط در یک نقطهٔ خاص متمرکز نیست. در هر لایه از معماری شبکه، باید نکات و موارد امنیتی مدّنظر قرار گرفته و بدقت رعایت شود: در لایهٔ فیزیکی (Physical Layer) برای جلوگیری از ایجاد انشعاب در سیم و پیشگیری از استراق سمع سیگنال (Wire tapping)، می‌توان سیمها را در درون یک لولهٔ محافظ جاسازی و لوله را با یک گاز تحت فشار پر کرد. در این حالت هر گونه تلاش در نقب زدن به این لوله و سوراخ کردن آن منجر به تخلیهٔ گاز، کاهش فشار لوله و به صدا در آمدن زنگهای هشدار خواهد شد. در برخی از سیستمهای نظامی از این روش استفاده شده است.

در لایهٔ پیوند داده‌ها (Data Link)، بسته‌های ارسالی بر روی خطوط نقطه به نقطه (Point To Point) قبل از خروج از ماشین مبداء، رمزنگاری شده و به محض ورود به ماشین مقصد رمزگشایی می‌شود. تمام جزئیات کار در سطح لایهٔ پیوند داده‌ها پیاده‌سازی و انجام می‌شود و لایه‌های فوقانی به آنچه که در این لایه اتفاق می‌افتد بی‌توجه هستند و بهیچوجه درگیر جزئیات آن نخواهند شد. این راه‌حل در مواقعی که بسته‌های حامل داده مجبور به گذر از چند مسیریاب باشند کارآیی خود را از دست خواهد داد زیرا اطلاعات در بدو ورود به هر مسیریاب رمزگشایی شده و بمحض خروج از آن مجدداً رمزنگاری می‌شوند فلذا این خطر وجود دارد که در یکی از مسیریابهای بین راه به اطلاعات حمله شود. از طرف دیگر با این روش نمی‌توان از نشستها (Sessions) حفاظت و مراقبت کرد (به عنوان مثال مراقبت از یک نشست Online برای خرید از طریق کارت اعتباری ممکن نیست) و در هر یک از مسیریابهای واقع برروی مسیر احتمال دستکاری یا استراق سمع اطلاعات وجود خواهد داشت. علیرغم این کمبودها، روش «رمزنگاری لینک» (Link Encryption) را می‌توان به سادگی به هر شبکه افزود و بیشتر مواقع نیز سودمند است.

در لایهٔ شبکه (Network Layer) می‌توان برای نظارت مؤثر بر ورود و خروج بسته‌های مجاز و تشخیص بسته‌های غیرمجاز (حامل داده‌های مخرب)، «**دیوار آتش**» (Firewall) نصب کرد. در ضمن در این لایه می‌توان از پروتکل IPsec (IP Security) استفاده کرد.

در لایهٔ انتقال (Transport Layer) کل اتصال (Connection) بین مبداء و مقصد، می‌تواند رمزنگاری شود. به عبارت بهتر تمام داده‌های در حال تبادل، در پروسهٔ مبداء رمز شده و در پروسهٔ مقصد از رمز خارج خواهد شد؛ (به این روش امنیت انتها به انتها یا End-to-End گفته می‌شود). برای تضمین حداکثر امنیت، إعمال «**امنیت انتهابه‌انتها**» الزامی است.

در آخر، مواردی نظیر احراز هویت کاربران و غیرقابل انکار بودن محتوای پیامها (Nonrepudiation) فقط در لایهٔ کاربرد قابل اعمال و پیاده‌سازی است.

از آنجایی که تضمین امنیت اطلاعات دقیقاً در یک لایهٔ خاص قرار نمی‌گیرد لذا نمی‌شد مفاد آنرا در ادامهٔ هیچیک از فصول این کتاب گنجاند؛ بهمین دلیل یک فصل اختصاصی و مجزا به آن اختصاص داده‌ایم.

این فصل طولانی، فنی و بنیادی است و همچنین ممکن است در نگاه اول اندکی گسیخته و نامربوط به نظر

برسد. بررسی مستندات به خوبی نشان می‌دهد که بسیاری از شکستها در حریم امنیتی شبکه‌هایی مثل بانکها، بیشتر ناشی از عواملی نظیر بی‌لیاقتی و سهل‌انگاری کارمندان، تمهیدات و روالهای امنیتی ناکافی، تقلب و کلاه‌برداریهای داخلی بوده است تا آنکه از یک نقشه جنایتکارانهٔ دقیق ، ایجاد انشعاب در خطوط تلفن ، سرقت و رمزگشایی اطلاعات منشاء گرفته باشد. اگر شخصی که یک کارت عابربانک (ماشین خودپرداز ATM) را در خیابان پیدا می‌کند براحتی بتواند به یکی از شعب مربوطه مراجعه نماید و با اعلام آنکه شمارهٔ PIN (شمارهٔ رمز شخصی) را فراموش کرده در همان شعبه شمارهٔ جدیدی دریافت کند (تحت عنوان ارتباط خوب و دوستانه با مشتریان!) آنگاه استفاده از الگوهای رمزنگاری و مکانیزمهای امنیتی، دیگر هیچ خاصیت و کاربردی در حفظ امنیت اطلاعات نخواهند داشت. کتاب Ross Anderson در این خصوص حقیقتاً هوشیارکننده و هشداردهنده است؛ مستندات کتاب مذکور نشان می‌دهد که در صدها نمونه از شکستهای امنیتی و نقض حریم شبکه‌ها در صنایع و سازمانهای متعدد، تقریباً تمام آنها (در یک عبارت مودبانه) ناشی از سهل‌انگاری در عملکرد و بی‌دقتی در رعایت نکات امنیتی بوده است. (Anderson, 2001) علیرغم این مسئله، خوشبین هستیم که با گسترش روزافزون تجارت الکترونیکی، موسسات و شرکتها در روالهای عملیاتی خود تجدیدنظر کرده، شکافهای امنیتی سیستمهای خود را برطرف نموده و جنبه‌های فنی «امنیت» را سرلوحه کار خود قرار بدهند.

به غیر از امنیت در سطح لایه فیزیکی (که به صورت مکانیکی و از طریق لوله‌های محتوی گاز تحت فشار انجام می‌شود) در بقیهٔ لایه‌ها تقریباً تمام مکانیزمهای امنیتی مبتنی بر اصول رمزنگاری است. به همین دلیل مطالعات خود در خصوص امنیت را با تشریح مبسوط مبانی رمزنگاری آغاز خواهیم کرد. در بخش ۱.۸ نگاهی بر اصول اولیهٔ آن خواهیم داشت. در بخش ۲.۸ تا بخش ۵.۸ به برخی از الگوریتمهای اساسی و ساختمان داده‌های مورد استفاده در رمزنگاری خواهیم پرداخت. سپس به صورت مبسوط تشریح خواهیم کرد که چگونه این مفاهیم برای تأمین امنیت در شبکه‌های کامپیوتری بکار گرفته می‌شوند. نهایتاً فصل را با خلاصه‌ای از نظریات در خصوص «تکنولوژی و جامعه» (Technology & Society) جمع‌بندی خواهیم کرد.

قبل از شروع، به مواردی که در این فصل بدانها نخواهیم پرداخت اشاره می‌کنیم. سعی کرده‌ایم در این فصل به مواردی در خصوص امنیت بپردازیم که مستقیماً در ارتباط با معماری شبکه است فلذا مواردی را که به سیستم عامل و برنامه‌های کاربردی مربوط می‌شود ، بررسی نکرده‌ایم. به عنوان مثال در این فصل مطلبی درخصوص احراز هویت کاربران به روش بیومتریک، استراتژیهای امنیت کلمهٔ عبور، **«حملهٔ از نوع سرریز کردن بافر»** و **«اسبهای تروا»** (Trojan Horses)، **«تقلب در ورود به سیستم»** (Login Spoofing)، بمبهای منطقی ، ویروسها، کرمها و نظایر آن وجود ندارد. تمام این موارد به تفصیل در فصل نهم از کتاب «سیستم عامل مدرن» (تاننباوم، ۲۰۰۱)بررسی شده‌اند. علاقمندان می‌توانند برای مطالعه بر روی این جوانب از امنیت، به کتاب فوق مراجعه نمایند. حال بیائید خط سیر خود را آغاز کنیم.

۱.۸ رمزنگاری

کلمهٔ Cryptography (رمزنگاری) برگرفته از لغات یونانی به معنای «محرمانه نوشتن متون» است. رمزنگاری پیشینه‌ای طولانی و درخشان دارد که به هزاران سال قبل بر می‌گردد. در این بخش برخی نکات برجسته از تاریخ رمزنگاری را بعنوان اطلاعات عمومی (که دانستن آنها برای ادامهٔ بخشهای بعد مفید خواهد بود)، بررسی خواهیم کرد. برای آشنایی با پیشینهٔ دقیق و کامل رمزنگاری، کتاب Kahn (۱۹۹۵) جهت مطالعه توصیه می‌شود؛ برای تحلیل جامع مفاهیم مدرن و پیشرفته در امنیت و آشنایی با الگوریتمهای رمزنگاری، پروتکلها و برنامه‌های کاربردی به کتاب Kaufman (۲۰۰۲) مراجعه کنید. برای آشنایی با جزئیات ریاضی این روشها کتاب Stinson

(۲۰۰۲) را ببینید. برای بررسی این روشها، بدون جزئیات ریاضی، به کتاب Burnett & Paine (۲۰۰۱) مراجعه کنید.

متخصصین رمزنگاری بین «رمز» (Cipher) و «کُد» (Code) تمایز قائل می‌شوند. «رمز» عبارتست از تبدیل کاراکتر به کاراکتر یا بیت به بیت بدون آن که به محتویات زبان شناختی (ادبیات) آن پیام توجه شود. در طرف مقابل، «کد» تبدیلی است که کلمه‌ای را با یک کلمه یا علامت (سمبول) دیگر جایگزین می‌کند. امروزه از کدها استفادهٔ چندانی نمی‌شود اگرچه استفاده از آن پیشینهٔ طولانی و پر سابقه‌ای دارد. موفقترین «کد»هایی که تاکنون ابداع شده‌اند توسط ارتش ایالات متحده و در خلال جنگ جهانی دوم در اقیانوس آرام بکار گرفته شد. آنها از لهجهٔ محلی Navajo در میان سرخپوستان الهام گرفته و برای عبارات و کلمات نظامی به سادگی از لغات خاص این زبان محلی استفاده کردند؛ به عنوان مثال عبارت chay-dagahi-nail-tsaidi (در زبان محلی سرخپوستان به معنای کُشنده‌لاک‌پشت!) رمزی برای سلاح ضد تانک بود. زبان Navajo لهجه‌ای بسیار آهنگین و بشدت پیچیده است و هیچ ادبیات نوشتاری و الفبای خطی ندارد و یک فرد ژاپنی (آن هم در جنگ جهانی دوم) هیچ چیزی در مورد آن نمی‌دانست.

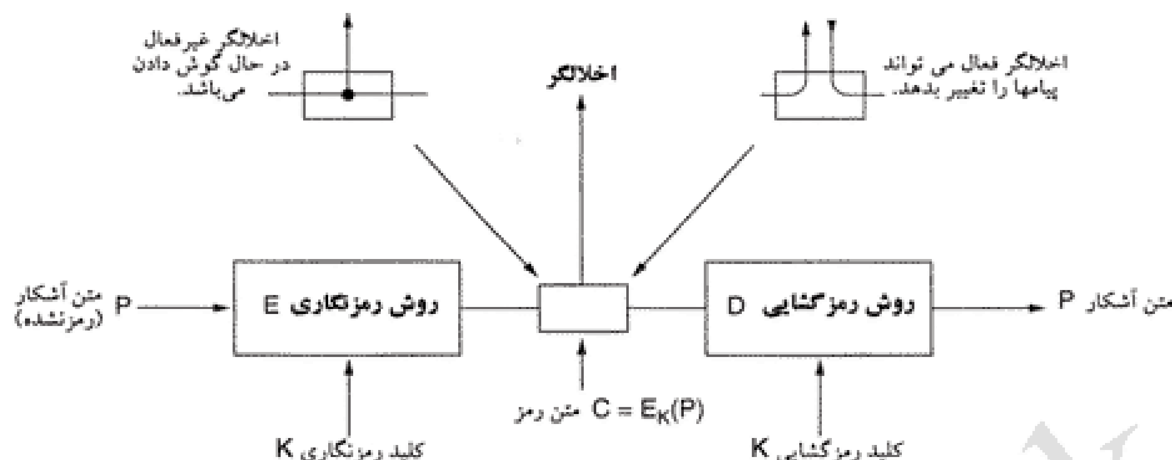
در سپتامبر ۱۹۴۵ در مجمع متفقین در سن‌دیه‌گو این «کُد» با بیان گزارش ذیل توصیف شد: «برای سه سال متوالی، هر گاه ناوگان دریایی در خشکی پهلو می‌گرفت، آنچه که جاسوسان ژاپنی (از بی‌سیم) استراق سمع می‌کردند یک صدای نامفهوم و شلوغ بود که با دیگر اصوات درهم آمیخته و در نتیجه صدایی شبیه به لحن راهبان تبت یا صدای یک بطری آبجوش که آب آن در حال خالی شدن باشد، می‌شنیدند!!» ژاپنی‌ها هرگز نتوانستند این کد را بشکنند؛ پس از جنگ بسیاری از افرادی که مبادلهٔ پیامهای سرّی جنگ را به زبان رمزی Navajo بر عهده داشتند بخاطر خدمات شایان و شجاعتشان در طول جنگ، مفتخر به دریافت نشانهای عالی نظامی شدند. ارتش ایالات متحده توانست کُدهای رمز ژاپنی‌ها را بشکند در حالی که ژاپنی‌ها نتوانستند کُد Navajo را بشکنند و این حقیقت، نقش بسیار مهمی در پیروزیهای آمریکاییان در جنگ اقیانوس آرام (با نیروی دریائی ژاپن) ایفاء کرد.

۸-۱-۱ مقدمه‌ای بر رمزنگاری

از دیدگاه تاریخ، چهار گروه از مردم در شکل‌گیری هنر رمزنگاری دخیل بوده‌اند: «نظامیان»، «هیئت‌های سیاسی»، «خاطره‌نویسان/واقعه‌نگاران» و «عشّاق!». از بین اینها نظامیان نقش بسیار مهمتری دارند و در طول قرنها به تکوین این شاخه از علم پرداخته‌اند. سابقاً در مؤسسات نظامی، پیامهائی که باید رمزنگاری می‌شدند به یک کارمند (منشی) دون پایه و حقوق‌بگیر تحویل می‌شد تا آنها را رمز و ارسال کند. حجم عظیم پیامهائی که در طی یک روز باید رمز و ارسال می‌شد مانع از آن بود که بتوان این کار خطیر را بر عهدهٔ معدود متخصصین خبرهٔ حاضر در یک مؤسسه گذاشت.

تا زمان ابداع کامپیوترها، در عرصهٔ یک جنگ واقعی و با تجهیزات اندک، بزرگترین نقطهٔ ضعف استراتژی رمزنگاری آن بود که همه چیز به توانائی و سرعت عمل کارمند رمزنگار پیام، وابسته و منوط می‌شد. محدودیت دیگر آن بود که نمی‌شد براحتی و سریع یک روش رمزنگاری را به روشی دیگر تغییر داد زیرا این کار مستلزم بازآموزی جمع کثیری از منشیان و کارمندان رمزنگار بود. از طرفی این خطر نیز وجود داشت که یکی از منشیان رمزنگار، دستگیر شده و روش رمزنگاری فاش گردد لذا باید این امکان مهیا می‌شد که به محض احساس لزوم، روش رمزنگاری تغییر کند. این مشکلات متناقض، منجر به پیدایش مدل شکل ۸-۲ شد.

پیامی که باید رمزنگاری شود، «**متن آشکار**» (Plaintext) نامیده می‌شود و توسط یک تابع خاص با پارامتری بنام «**کلید**» (Key) به متن رمز، تبدیل می‌گردد. نتیجهٔ فرآیند رمزنگاری که «**متن رمز**» (Ciphertext) نامیده می‌شود بر روی کانال منتقل خواهد شد. فرض کنیم که دشمن یا اخلالگر (Intruder) متن رمز شده را به صورت کامل



شکل ۸-۲. مدل رمزنگاری (برای روشهای رمزنگاری با کلید متقارن).

می‌شنود و آن را در اختیار می‌گیرد. به هر حال او برخلاف گیرندهٔ اصلی، براحتی قادر به رمزگشایی پیام و بهره‌برداری از آن نخواهد بود زیرا کلید رمز را نمی‌داند. برخی اوقات یک اخلالگر غیرفعال (Passive Intruder) نه تنها قادر است به جریان اطلاعات بر روی کانال مخابراتی گوش بدهد بلکه می‌تواند آنها را در جایی ثبت کرده و بعداً آن را بارها به جریان بیندازد؛ در مقابل یک اخلالگر فعّال (Active Intruder) می‌تواند پیام مورد نظر خود را در داخل یک پیام مجاز و معتبر جاسازی کند یا در آن دستکاری نماید. هنر شکستن رمز بدون در اختیار داشتن کلید آن ، «**علم تحلیل رمز**» (Cryptoanalysis) نام دارد؛ به هنر ابداع روشهای رمزنگاری جدید «**علم رمزنگاری**» (Cryptology) اطلاق می‌شود.

در اختیار داشتن یک نماد و فرمول ریاضی که ارتباط بین متن آشکار، متن رمز شده و کلید رمز را مشخص کند بسیار مفید خواهد بود. ما از نماد $C=E_K(P)$ استفاده خواهیم کرد، بدین معنا که عملیات رمزنگاری بر روی متن آشکار P توسط کلید رمز K انجام شده و متن رمزشدهٔ C بدست آمده است. به روش مشابه، فرمول $P=D_K(C)$ عمل رمزگشایی متن رمزشده توسط کلید K را (به منظور استخراج اصل پیام) توصیف می‌کند. بنابراین داریم:

$$D_K(E_K(P))=P$$

این نماد بیانگر آن است که E و D توابع ریاضی و معکوس یکدیگر هستند. تنها نکتهٔ قابل اشاره آنست که این توابع دارای دو پارامتر هستند، اگرچه کلید رمز K را که در حقیقت یکی از پارامترهای این توابع است به صورت پانویس برای E یا D نشان داده‌ایم تا تمایز آن از پیام مشخص باشد.

یکی از قواعد اساسی در علم رمزنگاری آن است که شخص باید فرض را بر آن بگذارد که دیگران [از جمله تحلیلگران رمز و رمزشکنها] الگوریتم بکار رفته در عملیات رمزنگاری را می‌دانند. به عبارت دیگر شخص رمزشکن، روش رمزنگاری یعنی تابع E و روش رمزگشایی یعنی تابع D در شکل ۸-۲ را می‌داند و آنچه از او پنهان نگاه داشته می‌شود فقط کلید رمز (K) است. میزان نیرو و تلاشی که باید برای ابداع، آزمایش و نصب یک الگوریتم رمزنگاری جدید (در صورت فاش شدن روش قبلی) انجام بگیرد بقدری زیاد است که محرمانه نگهداشتن روش رمزنگاری عملاً ممکن نیست. تصور آنکه الگوریتم رمزنگاری می‌تواند سری و مخفی بماند (در حالی که ممکن نیست) خطرات و زیانهائی بیشتر از منافع آن دارد.

اینجاست که «**کلید رمز**» وارد قضیه می‌شود. «کلید رمز» یک رشتهٔ کاراکتری نسبتاً کوتاه است که پیام براساس آن رمز می‌شود. برخلاف آن که روش رمزنگاری ممکن است هر چند سال یکبار تغییر کند، کلید رمز می‌تواند بر

طبق نیاز و به دفعات عوض شود. بنابراین مدل پایه سیستمهای رمزنگاری، مدلی است پایدار (ثابت) که همه از عملکرد و الگوریتم آن مطلعند و فقط با یک کلید محرمانه و قابل تغییر کار می‌کند. این نظریه که «تحلیل‌گر رمز» (رمزشکن / Cryptanalyst) از الگوریتم رمزنگاری آگاه است و سری ماندن یک پیام صرفاً به مخفی ماندن کلید رمز وابسته است «اصل کِرکهُف» نامیده می‌شود که توسط یکی از رمزنگاران ارتش فلاندرز به نام Auguste Kerckhoff در سال ۱۸۸۳ بیان شده است. بنابراین داریم:

***اصل کِرکهُف:* تمام الگوریتمهای رمزنگاری باید آشکار و عمومی باشند و تنها کلیدهای رمز، مخفی و محرمانه هستند.**

شاید نتوان حقّ این مطلب را که «تکیه بر مخفی ماندن الگوریتم رمزنگاری اشتباه است»، بدرستی ادا کرد. تلاش برای سری نگه داشتن الگوریتم رمزنگاری که در عُرف عامه به اصطلاح **«امنیت در سایهٔ گمنامی و ابهام»** (Security by Obscurity) مشهور است، هرگز محقق نخواهد شد. با عمومی‌سازی یک الگوریتم، طراح یک الگوریتم رمزنگاری می‌تواند از نیروی عظیم متخصصین رمزنگاری که مشتاق به شکستن یک سیستم هستند، مشورت بگیرد و آنها را به مبارزه بطلبد؛ آنها نیز می‌توانند در خصوص تلاشهایی که برای درهم شکستن یک سیستم رمزنگاری مصروف کرده‌اند مقاله بنویسند و خبرگی و هوش خود را به رخ بکشند! هر گاه متخصصین کثیری سعی در شکستن یک الگوریتم کردند و با گذشت پنج سال پس از انتشار عمومی آن هیچ گزارشی مبنی بر موفقیت آنان مشاهده نشد، آن الگوریتم احتمالاً بقدر کافی سخت و محکم بوده است!

از آنجایی که سری ماندن پیامها وابسته به کلید است لذا **طول کلید** یکی از نکات بسیار مهم در طراحی الگوریتمهای رمزنگاری است. به عنوان مثالی ساده، یک قفل رمزدار ترکیبی [مثل قفل بعضی از کیفهای شخصی] را در نظر بگیرید. قاعدهٔ کلی برای باز شدن این قفل آن است که چند رقم را به ترتیب وارد کنید. همه این موضوع [الگوریتم] را می‌دانند و لیکن کلید رمز محرمانه است. کلید رمز دو رقمی تنها صد حالت مختلف دارد. کلید سه رقمی معادل با هزار و کلید شش رقمی معادل یک میلیون حالت مختلف است. هر چه طول یک کلید بزرگتر باشد، حجم عملیات سعی و خطائی که رمزشکن برای دسترسی به کلید رمز باید انجام بدهد زیادتر خواهد بود.[۱] حجم عملیات (Work Factor) برای شکستن یک سیستم رمز از طریق آزمون تمام فضای حالات کلید، برحسب طول کلید به صورت نمائی رشد خواهد کرد. سری ماندن و امنیت پیامها با داشتن یک الگوریتم بسیار قدرتمند (ولی آشکار و همگانی) به همراه یک کلید طولانی تضمین می‌شود. برای آن که نگذارید برادر کوچک شما نامه‌های الکترونیکی شما را بخواند یک کلید ۶۴ بیتی (هشت کاراکتری) کفایت می‌کند. برای انجام عملیات معمول اقتصادی باید از کلیدی با حداقل ۱۲۸ بیت استفاده شود. برای حراست از پیامهای سری دولتی به کلیدهایی با طول حداقل ۲۵۶ بیت یا حتی بیشتر نیاز است.

از دیدگاه یک تحلیل‌گر رمز (رمزشکن)، مسئله کشف رمز سه شِق اساسی را در بر می‌گیرد: (۱) هر گاه او فقط توده‌ای از متن رمز شده (بدون هیچ متن آشکار و بدون کلید) در اختیار داشته باشد با مسئله‌ای به نام **«صرفاً متن رمز شده»**[۲] مواجه است. (۲) وقتی رمزشکن بخشی از متن آشکار را به همراه معادل رمز شدهٔ آن، در اختیار دارد، اصطلاحاً با مسئله **«متن آشکار و شناخته شده»**[۳] مواجه است. (۳) نهایتاً هر گاه رمزشکن قادر باشد هر قسمت دلخواه از یک متن آشکار را رمز کند اصطلاحاً با مسئله **«متن آشکار و انتخابی»**[۴] مواجه است. هرگاه به یک رمزشکن اجازه داده شود تا بپرسد مثلاً حاصل رمزنگاری رشته ABCDEFGHIJKL چیست به سادگی قادر

---

۱. به حجم عملیات لازم برای کشف کلید رمز به روش سعی و خطا، اصطلاحاً Work Factor گفته می‌شود. -م.

۲. Ciphertext Only ۳. Known Plaintext ۴. Chosen Plaintext

خواهد بود تا رمزهای معمولی را بشکند و کلید رمز را بدست بیاورد.

نوآموزان حرفهٔ رمزنگاری به اشتباه می‌اندیشند که هرگاه یک متن رمز شده بتواند در مقابل حملهٔ نوع «صرفاً متن رمز شده» استقامت کند و فاش نشود، آن سیستم رمز مطمئن است.[۱] این فرض کاملاً خام و ناشیانه است زیرا در بسیاری از حالات، رمزشکن قادر است حدسهای درست و موثقی را در مورد برخی از قسمتهای یک متن رمز شده آزمایش کند.[۲] به عنوان مثال اولین جمله‌ای که در حین برقراری ارتباط شخص از راه دور [مثلاً با یک سرویس‌دهنده TelNet] ، ارسال می‌شود معمولاً کلمهٔ :login (به صورت رمز شده) است. حال رمزشکن پاره‌ای متن رمز شده به همراه معادل رمز نشدهٔ آن را در اختیار دارد؛ کار او نسبتاً ساده و سرراست است. برای رسیدن به امنیت کامل، طراح الگوریتم رمزنگاری باید محافظه‌کار بوده و مطمئن شود که سیستم رمز او بگونه‌ای است که حتی در صورتی که حریف رمزشکن ، معادل رمز شدهٔ یک متن آشکار را در اختیار داشته باشد باز هم قادر به شکستن رمز و بدست آوردن کلید رمز نیست.[۳]

روشهای رمزنگاری بطور کلی به دو رده تقسیم می‌شوند: (۱)رمزهای جانشینی (Substitution) (۲)رمزهای جایگشتی (Transposition) . در اینجا به عنوان مقدمه‌ای بر روشهای مدرن رمزنگاری، مختصراً به این دو روش خواهیم پرداخت.

۸-۱-۲ رمزهای جانشینی (Substitution Cipher)

در رمزنگاری جانشینی هر حرف یا گروهی از حروف با یک حرف یا گروهی دیگر از حروف جابجا می‌شوند تا شکل پیام بهم بریزد. یکی از قدیمی‌ترین رمزهای شناخته شده روش رمزنگاری سزار است که ابداع آن به ژولیوس سزار نسبت داده می‌شود. در این روش حرف a به D تبدیل می‌شود، b به E ، c به F و به همین ترتیب تا z که با C جایگزین می‌گردد. به عنوان مثال در این روش، کلمهٔ attack به DWWDFN تبدیل می‌شود. در مثالها متن پیام متشکل از حروف کوچک فرض شده و متن رمز شده صرفاً شامل حروف بزرگ انگلیسی است.

یک حالت عمومی و ساده از رمزنگاری سزار آن است که هر حرف الفباء از متن اصلی با حرفی که در جدول الفباء k حرف بعدتر قرار گرفته جابجا شود. (روش *Shift by k*) در این روش کلید رمز عدد k خواهد بود و براساس آن حروف یک متن به صورت چرخشی (Circular) با حرف kام بعد از خودش جایگزین می‌شود. روش رمزنگاری سزار شاید در آن زمانها کسی را فریفته باشد ولی امروزه نمی‌تواند هیچکس را گول بزند. بهبود بعدی این روش آن است که هر حرف در متن اصلی با یک حرف دلخواه جانشین شود، یعنی ۲۶ حرف جدول الفباء به حروف دیگری در همان جدول نگاشته شود. به عنوان مثال از نگاشت زیر می‌توان برای رمزنگاری جانشینی استفاده کرد:

متن آشکار a b c d e f g h i j k l m n o p q r s t u v w x y z
متن رمز شده Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

هر سیستم رمزنگاری که در آن یک سمبول با سمبول دیگر جایگزین می‌شود اصطلاحاً «**سیستم جانشینی**

---

۱. بدین خیال که چون رمزشکن فقط متن رمز شده را در اختیار دارد و چیزی از متن اصلی نمی‌داند بنابراین قادر به آزمون تمام حالات مختلف کلید رمز نیست و بالطبع کلید فاش نخواهد شد. -م

۲. به عبارت دیگر اگرچه به زعم دیگران او از متن رمزشده چیزی نمی‌داند ولی رمزشکن قادر است بخشهای کوچکی از یک متن را حدس بزند. م

۳. بعبارت روشنتر سیستم رمز باید بنحوی طراحی شود که حتی اگر یک متن رمزشده و معادل رمزنشدهٔ آنرا به رمزشکن بدهید بازهم نتواند کلید رمز شما را پیدا کند. -م

تک‌حرفی» (Monoalphabetic Substitution) گفته می‌شود که در آن کلید رمز یک رشتهٔ ۲۶ کاراکتری است و نگاشت جدول الفباء را مشخص می‌نماید. بر اساس کلید رمزِ مثال بالا، کلمهٔ attack به QZZQEA تبدیل خواهد شد.

در نگاه اول این سیستم رمزنگار مطمئن به نظر می‌رسد زیرا اگرچه رمزشکن روش عمومی جانشینی حروف را می‌داند ولی نمی‌داند از بین !26 حالت مختلف (معادل با 4×10^{26} حالت) کدامیک کلید رمز است. برخلاف رمز سزار، آزمایش تمام حالات مختلف کلید غیرممکن است زیرا اگر هر یک از حالات کلمهٔ رمز در یک نانوثانیه آزمایش شود، بررسی تمام حالات کلید توسط چنین کامپیوتری 10^{10} سال طول خواهد کشید.

در روش فوق علیرغم آنکه آزمایش تمام حالات یک کلید ممکن نیست ولی حتی برای یک قطعه متن رمز شدهٔ کوچک، رمز متن به سادگی شکسته خواهد شد. در حملهٔ اصولی به این سیستم رمز از ویژگیهای آماری زبانهای طبیعی بهره گرفته شده است. به عنوان مثال در زبان انگلیسی حرف ***e*** بیشترین تکرار را در متون معمولی دارد؛ به دنبال آن حرف ***t***، سپس ***o***، ***a***، ***n*** و ***i*** در رتبه‌های بعدی قرار می‌گیرند. ترکیبات دو حرفی که اصطلاحاً digram نامیده می‌شوند به ترتیب بیشترین تکرار عبارتند از: (۱) ***th*** (۲) ***in*** (۳) ***er*** (۴) ***re*** (۵) ***an*** و بهمین ترتیب. ترکیبات سه حرفی حروف انگلیسی (Trigram) به ترتیب بیشترین تکرار عبارتند از: (۱) ***the*** (۲) ***ing*** (۳) ***and*** و (۴) ***ion***

تحلیل‌گر رمز (رمزشکن) برای شکستن سیستم رمزنگاری «جانشینی تک‌حرفی» (Monoalphabetic) با شمارش حروف متن رمز شده و محاسبهٔ تکرار نسبی هر حرف شروع می‌کند؛ سپس حرفی را که دارای بیشترین تکرار است با ***e*** و حرف پر تکرار بعدی را با ***t*** جایگزین می‌کند. حال او می‌تواند با در نظر داشتن سه حرفی ***the*** به دنبال سه حرفی‌های ***tXe*** در متن رمز شده بگردد.؛ به احتمال قوی ***X*** معادل با ***h*** است. سپس به روش مشابه به سراغ چهار حرفی‌های ***thYt*** می‌گردد. به احتمال زیاد ***Y*** معادل با ***a*** است. با اطلاعاتی که بدست آمده است او به دنبال سه حرفیهای مکرر با الگوی ***aZW*** می‌گردد که احتمالاً معادل با ***and*** خواهد بود. با حدس زدن بقیهٔ یک‌حرفیها، دوحرفیها و سه‌حرفیهای تکراری و با شناخت از حروف صدادار و بی‌صدا و چگونگی ترکیب آنها در کلمه، رمزشکن می‌تواند متن اصلی را به روش سعی و خطا (حرف به حرف) بدست آورد.

یک روش دیگر برای شکستن رمز متون آن است که یک کلمه یا یک عبارت کامل حدس زده شود. به عنوان مثال به متن رمز شدهٔ زیر که متعلق به یک شرکت حسابرسی است (و به صورت دسته‌های پنج کاراکتری نشان داده شده است) دقت کنید:

```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX  VSZJU JDSTS  JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ  BNYBN QJSW
```

احتمال وجود کلمهٔ «financial» در پیامهای یک شرکت حسابرسی زیاد است. با استفاده از این ویژگی که کلمهٔ financial دارای دو حرف «***i***» با فاصلهٔ چهار حرف از یکدیگر است، در متن رمزشده به جستجوی حروف مشابه با فاصلهٔ چهار حرف از یکدیگر می‌پردازیم. با جستجو در متن فوق به ۱۲ مورد تطابق در موقعیتهای: ۶، ۱۵، ۲۷، ۳۱، ۴۲، ۴۸، ۵۶، ۶۶، ۷۰، ۷۱، ۷۶ و ۸۲ بر می‌خوریم. ولیکن از بین این دوازده مورد فقط موارد ۳۱ و ۴۲ هستند که در آنها حرف بعدی (متناظر با ***n*** از کلمهٔ financial) با فاصلهٔ یک حرف تکرار شده و مطابقت دارد. از بین این دو مورد نیز فقط مورد ۳۱ است که با تکرار حرف ***a*** با فاصلهٔ سه حرف مطابقت دارد؛ لذا می‌توان متوجه‌شد که در این متن کلمهٔ financial در موقعیت ۳۰ از متن رمز شده قرار گرفته است. پیدا شدن این کلمه، نقطهٔ شروع خوبی برای پیدا کردن کلید رمز با کمک ویژگیهای آماری حروف در زبان انگلیسی خواهد بود.

۸-۱-۳ رمزنگاری جایگشتی (Transposition)

رمزنگاری جانشینی ترتیب سمبولهای یک متن را حفظ می‌کند ولی (صرفاً) شکل سمبولها را تغییر می‌دهد. برعکس، «رمزنگاری جایگشتی» ترتیب حروف متن را بهم می‌ریزد ولیکن شکل آنها را تغییر **نخواهد داد**. در شکل ۸-۳ یک روش عمومی از رمزنگاری جایگشتی که در آن ترتیب سمبولها بصورت ستونی بهم ریخته می‌شود نشان داده شده است. کلید رمز یک کلمه یا عبارت [انگلیسی] است که هیچ حرف تکراری ندارد. در این مثال کلید رمز کلمهٔ MEGABUCK انتخاب شده است. کاربرد کلید رمز آنست که ستونها شماره‌گذاری شود. شمارهٔ هر ستون براساس ترتیب الفبایی هر حرف کلید نسبت به جدول الفبای انگلیسی تعیین می‌شود. به عنوان مثال ستون چهارم شمارهٔ ۱ است (حرف A)، ستون پنجم شماره ۲ (حرف B)، ستون هفتم شمارهٔ ۳ (حرف C) و ستون دوم شمارهٔ ۴ (حرف E) و به همین ترتیب. متن اصلی به صورت افقی (سطری) در ذیل کلید رمز نوشته می‌شود. سپس در صورت لزوم تعدادی حرف به آخرین سطر اضافه می‌شود تا ماتریس مربوطه پر شود. متن رمز شده بر اساس شمارهٔ ستونها به صورت عمودی خوانده شده و به هم متصل می‌شود. ترتیب خواندن ستونها، از ستون با کمترین شماره به بزرگترین شماره است.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

متن آشکار

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

متن رمزشده

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

شکل ۸-۳. رمزنگاری جایگشتی.

برای شکستن رمز فوق، تحلیل‌گر رمز ابتدا باید مطمئن شود که آیا واقعاً با یک متن رمز شده به روش جایگشت روبرو است یا خیر؟ با بررسی تکرار حروف E، T، A، O، I، N و نظائر آن بسادگی می‌توان مطمئن شد که متن الگوئی شبیه به یک متن معمولی رمزنشده دارد. در این صورت روشن است که متن رمز شده از نوع جایگشتی است زیرا در این روش شکل هر حرف تغییر نمی‌کند بلکه فقط جای آن در متن عوض می‌شود و این کار در آمار حروف و میزان تکرار آنها تأثیری ندارد.

گام بعدی آن است که تعداد ستونها حدس زده شود. در بیشتر مواقع می‌توان برخی از کلمات یا عبارات یک متن را حدس زد. به عنوان مثال فرض کنید رمزشکن ما به وجود عبارت milliondollars در جایی از متن اصلی مشکوک است. دقت کنید که دو حرفی‌های MO، IL، LL، LA، IR و OS در متن رمز، دیده می‌شود که ناشی از شکسته و چیده شدن عبارت فوق به صورت ستونی است. حرف O در متن رمز شده پشت M ظاهر شده است. (یعنی در اثر چیده شدن عبارت فوق در سطرها، این دو حرف در ستون چهارم کنار هم قرار گرفته‌اند.) «حروفی که در متن رمز شده کنار هم قرار می‌گیرند در متن اصلی به اندازه «طول کلید» از هم فاصله دارند». اگر طول کلید به جای ۸، هفت در نظر گرفته می‌شد، پس از رمز شدن عبارت milliondollars ، دو حرفی‌های MD، IO، LL، LL، IA، OR و NS بوجود می‌آمدند. در حقیقت بسته به طول کلید مجموعهٔ متفاوتی از دو حرفی‌ها در متن رمز شده ظاهر می‌شود. تحلیل‌گر رمز می‌تواند با آزمایش طولهای مختلف کلید، به سادگی طول کلید را بدست بیاورد.

گام آخر، بدست آوردن ترتیب ستونها است. وقتی تعداد ستونها (k) کم باشد به سادگی می‌توان تمام $k\times(k-1)$ حالت مختلف زوج ستونها را آزمود تا ببینیم آیا میزان تکرار دو حرفیها و سه حرفیهای این دو ستون با شرایط آماری یک متن معمولی مطابقت دارد یا خیر؟ حالتی از ترکیب ستونها که براساس آن ترکیب، نتیجهٔ رمزگشایی بیشترین تطبیق را با متون معمولی داشته باشد به عنوان زوج ستون متوالی در نظر گرفته می‌شود. حال یک‌ایک ستونها به عنوان ستونهای بعدی این زوج رمزگشایی شده، آزمایش و تحلیلهای آماری بر روی آنها انجام می‌شود و بهترین ستون به عنوان ستون بعدی آن در نظر گرفته می‌شود و این کار ادامه می‌یابد. ستون قبلی یک ستون نیز به همین روش (تحلیل دو حرفی و سه حرفیها) امکان‌پذیر است. این فرآیند آنقدر تکرار می‌شود تا ترکیب درست و مورد تائید به دست بیاید. عمل رمزشکنی در نقطه‌ای با اقبال و موفقیت روبرو خواهد شد که یک کلمه یا یک عبارت ظاهر گردد. (به عنوان مثال هر گاه در آزمون یک ترکیب خاص از ستونها، کلمهٔ million ظاهر شود می‌توان ترتیب واقعی ستونها و اشتباهاتی که قبلاً در حدسها وجود داشته را روشن کرد.)

برخی از سیستمهای رمزنگاری جایگشتی، یک بلوک از کاراکترها با طول ثابت را از ورودی دریافت کرده و یک بلوک رمز شده (با طول ثابت) در خروجی تولید می‌کنند. در این گونه از روشها فهرست کامل جایگشتهای ورودی (که متن رمز شدهٔ خروجی را تولید می‌کند) مشخص است. به عنوان مثال سیستم رمز شکل ۸-۳ را می‌توان در قالب یک رمزنگار با بلوکهای ورودی ۶۴بایتی تصور کرد که فهرست جایگشتها عبارتست از ۴، ۱۲، ۲۰، ۲۸، ۳۶، ۴۴، ۵۲، ۶۰، ۵، ۱۳، ۲۱ ،... و ۶۲. به عبارت دیگر چهارمین کاراکتر ورودی (یعنی a) اولین کاراکتر خروجی این رمزنگار خواهد بود.

۸-۱-۴ رمز One-Time Pads (بهم ریزی محتوی پیام)

ایجاد یک سیستم رمزنگاری که غیرقابل شکستن باشد حقیقتاً کار ساده‌ایست و روش آن نیز از دهها سال قبل شناخته شده است: ابتدا یک رشته بیت تصادفی را به عنوان کلید انتخاب کنید. سپس متن اصلی (رمز نشده) را به یک رشته بیت متوالی تبدیل نمائید؛ (مثلاً با الحاق بیتهای کد اَسکی هر کاراکتر) نهایتاً این دو رشته را بیت به بیت با یکدیگر XOR (Exclusive OR) کنید. رشته بیت حاصل، متن رمز شدهٔ شماست که هرگز قابل شکستن نیست زیرا در صورتی که متن رمز شده به قدر کافی بزرگ باشد هر حرف در این متن به یک نسبت تکرار خواهد شد، همچنین میزان تکرار تمام دوحرفیها و سه‌حرفیها (Digram/Trigram) در متن رمز، مشابه خواهد بود.[۱] این روش که به نام One-Time Pad مشهور است در برابر تمام حملات فعلی یا حملات احتمالی آینده مصون خواهد ماند و میزان توان محاسباتی و هوش رمزشکن هیچ تأثیری در شکستن آن نخواهد داشت. دلیل منطقی شکست ناپذیری این روش رمزنگاری، از «**تئوری اطلاعات**» (Information Theory) استنتاج می‌شود: در صورت انتخاب کلید کاملاً تصادفی، هیچ اطلاعاتی از پیام اصلی در پیام رمز شده باقی نخواهد ماند زیرا تمام حروف و سمبولها با احتمال وقوع مشابه در متن رمز شده تکرار خواهند شد.

مثالی از چگونگی رمزنگاری در روش One-Time Pad در شکل ۸-۴ مشاهده می‌شود. ابتدا پیام ۱ (جملهٔ "I Love You") کاراکتر به کاراکتر به کدهای اسکی هفت‌بیتی تبدیل می‌شوند. سپس یک کلید رمز تصادفی [که از این به بعد آن را Pad می‌نامیم] انتخاب و با پیام XOR می‌شود تا متن رمز شده بدست آید. یک رمزشکن باید تمام حالات مختلف رشتهٔ Pad را آزمایش کند تا ببیند به ازای هر Pad چه متنی حاصل می‌شود که البته باز هم موفق به یافتن متن اصلی نخواهد شد زیرا به عنوان مثال اگر Pad شمارهٔ ۲ با پیام اول XOR شود رشته‌ای را حاصل خواهد

---

۱. به عبارت فنی و دقیق‌تر اگر کلید رمز مناسب انتخاب شود متن رمز شده هیچیک از خصوصیات آماری یک متن معمولی را نخواهد داشت و رمزشکن هیچ راهی برای تحلیل متن و کشف رمز نخواهد داشت. -م

کرد که آن هم متن معمولی و معادل با متن "Elvis Lives" است و احتمالاً به عنوان متن واقعی تلقی و باور خواهد شد که البته اشتباه است. به عبارت بهتر برای هر متن یازده کاراکتری مثال فوق، یک Pad وجود دارد [که پس از XOR شدن با متن رمز] عبارت Elvis Lives (یا هر متن دلخواه دیگر) را تولید خواهد کرد. بنابراین وقتی می‌گوییم که یک متن پس از XOR شدن با یک Pad دلخواه، در خصوص متن اصلی هیچ اطلاعاتی در خود ندارد منظورمان آن است که می‌توانید از متن رمز شده هر پیامی به غیر از پیام اصلی و با طول مشابه استخراج کنید.

پیام ۱ :	1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Pad 1:	1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
متن رمز :	0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101
Pad 2:	1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
متن آشکار ۲ :	1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

شکل ۸-۴. استفاده از روش One-Time Pad برای رمزنگاری و این امکان که می‌توان بکمک برخی از Padها هر متن دلخواهی را از این رشتهٔ رمز استخراج کرد.

روش رمزنگاری One-Time Pad اگرچه از دیدگاه تئوری عالی و امن به نظر می‌رسد ولیکن در عمل با اشکالات عمده‌ای مواجه است. به عنوان اولین اشکال، کلید را نمی‌توان بخاطر سپرد و هم گیرنده و هم فرستندهٔ پیام، باید آنرا به صورت نوشته شده با خود حمل کنند. اگر یکی از این دو طرف در معرض حملهٔ فیزیکی یا سرقت کلید قرار داشته باشند، کلیدهایی که در جایی یادداشت شده هرگز مطلوب و قابل اعتماد نخواهد بود. همچنین حجم کل داده‌هایی که می‌تواند ارسال شود به طول کلید مورد استفاده بستگی دارد. مشکل دیگر در این روش، حساسیت به کاراکترهای جا افتاده (گمشده) یا اضافی است زیرا اگر یک کاراکتر از درون متن حذف یا به درون آن اضافه شود از آن محل به بعد، متن قابل رمزگشایی نخواهد بود لذا اگر به هر دلیلی گیرنده و فرستنده هماهنگی خود را از دست بدهند (یا به عبارت بهتر از حالت سنکرون خارج شوند) از آن لحظه به بعد تمام داده‌های رمزگشایی‌شده غیرقابل استفاده و آشغال خواهد بود.

با ابداع کامپیوترها، ممکن است استفاده از روش رمزنگاری One-Time Pad در برخی از برنامه‌های کاربردی امکان‌پذیر و عملی باشد. (به عنوان مثال) کلید می‌تواند بر روی یک DVD خاصّ حاوی چندین گیگابایت اطلاعات، ذخیره گردد؛ اگر کلید رمز در پوشش چند دقیقه فیلم ویدیویی بر روی DVD جاسازی شود شک‌برانگیز نخواهد بود. البته در شبکه‌هایی با سرعت گیگابیت بر ثانیه تغییر DVD مثلاً در هر سی ثانیه (بمنظور تغییر کلید رمز) کاری ملال‌آور و غیرممکن است. در ضمن DVDهای حامل کلیدهای رمز باید توسط شخص فرستندهٔ اطلاعات و قبل از ارسال داده‌ها، به گیرنده تسلیم شود، که این کار بشدت از عملی بودن روش خواهد کاست.[۱]

رمزنگاری کوآنتومی

راه حل جالبی برای مشکل انتقال کلید رمز به روش One-Time Pad بر روی شبکه وجود دارد؛ ابداع این راه حل منشاء عجیب و دور از ذهنی دارد: «مکانیک کوآنتومی»! هر چند این روش هنوز در مراحل تستهای آزمایشگاهی بسر می‌برد ولی آزمایشات اولیه امیدوارکننده بوده است. اگر این روش بتواند تکمیل شده و کارآیی آن تضمین گردد، در آینده احتمالاً تمام سیستمهای رمزنگاری براساس روش One-Time Pad شکل خواهد گرفت زیرا این

---

۱. بزرگترین مشکل روش رمزنگاری One-Time Pad را طولانی بودن طول کلید و حمل و نقل آن، در نظر بگیرید. -م

روش کاملاً امن و غیرقابل شکستن است. در زیر بطور مختصر توضیح خواهیم داد که روش رمزنگاری کوآنتومی چگونه کار می‌کند. بطور خاص پروتکل BB84 را بررسی خواهیم کرد. (BB84 ابتدای نام ابداع‌کنندگان و سال انتشار مقالهٔ آنهاست ـ Bennet and Brassard ، ۱۹۸۴)

یک کاربر مثل آلیس مایل است با کاربر دوم (مثلاً باب) یک Pad طولانی ایجاد و از آن به عنوان کلید رمز استفاده کند. آلیس و باب که اصطلاحاً طرفین اصلی ـ Principals ـ نامیده می‌شوند بازیگران اصلی داستان ما هستند. به عنوان مثال باب یک بانکدار و آلیس یکی از مشتریان بانک است که می‌خواهد با باب مراودات تجاری داشته باشد. در چند دههٔ گذشته در مقالات و کتب رمزنگاری، اسامی «آلیس» و «باب» عموماً به عنوان طرفین مُجاز و بازیگران اصلی سناریوهای امنیتی انتخاب شده است. رمزنگارها به سنت پایبند هستند! اگر به جای این دو نام، نامهای آندی و باربارا را به عنوان بازیگران سناریومان انتخاب می‌کردیم هیچکس مطالب این فصل را باور نمی‌کرد! پس بگذارید ما هم از این سنّت تبعیت کنیم.

اگر آلیس و باب بتوانند یک Pad طولانی به عنوان کلید رمز، ایجاد کنند، قادرند با استفاده از آن ، داده‌ها را به صورت مطمئن و امن مبادله نمایند. سؤال آن است که آنها چگونه می‌توانند بدون ردّ و بدل کردن فیزیکی کلید رمز (مثلاً با DVD)، Padها را ایجاد و مبادله کنند؟ می‌توانیم فرض کنیم که آلیس و باب در دو سمت یک فیبرنوری قرار گرفته‌اند و می‌توانند پالسهای نوری را ارسال یا دریافت نمایند ولیکن به فرض یک متجاوز به نام ترودی توانسته فیبرنوری را قطع کرده و در آن یک انشعاب فعال ایجاد کند. بدین ترتیب ترودی قادر است تمام بیتها را در دو جهت استراق سمع نماید. او همچنین می‌تواند پیامهای غلط برای دو طرف ارسال کند. این شرایط اگرچه برای باب و آلیس ناامیدکننده و خطرناک به نظر می‌رسد ولیکن رمزنگاری کوآنتومی می‌تواند روزنهٔ امیدی برای حل این مشکل باشد.

رمزنگاری کوآنتومی بر این اصل استوار است که نور در قالب بسته‌های کوچکی به نام فوتون با ویژگیهای خاصّ و عجیب، جابجا می‌شود. به علاوه وقتی نور از یک فیلتر پلاریزه‌کننده عبور می‌کند، پلاریزه (قطبی) می‌شود. نور پلاریزه شده برای عکاسان یا آنهایی که عینک آفتابی می‌زنند بخوبی ملموس است. اگر یک پرتو نوری (یا به عبارتی جریان فوتونها) از یک فیلتر پلاریزه کننده عبور کند، تمام فوتونهای پرتو خارج شده از فیلتر، در راستای محور فیلتر (محور عمودی) پلاریزه می‌شوند. حال اگر این پرتو مجدداً از فیلتر پلاریزه‌کنندهٔ دوم عبور نماید «شدت نور» پرتوی خروجی، متناسب با مربع کسینوس زاویهٔ بین محورهای عمودی دو فیلتر ($Cos^2\varphi$) خواهد بود. اگر محورهای دو فیلتر بر هم عمود باشند هیچ یک از فوتونها از بین این دو فیلتر عبور نخواهند کرد. البته زاویهٔ مطلق دو فیلتر در فضای سه بعدی اصلاً مهم نیست بلکه فقط زاویهٔ نسبی محورهای دو فیلتر پلاریزه کننده در محاسبه وارد می‌شود.

برای تولید یک Pad به عنوان کلید رمز، آلیس نیاز به تنظیم دو زوج فیلتر پلاریزه کننده دارد. زوج فیلتر اول، شامل یک فیلتر عمودی و یک فیلتر افقی است. این انتخاب اصطلاحاً **«دستگاه مستقیم»** (Rectilinear Basis) نامیده می‌شود. زوج فیلتر دوم مشابه با قبلی است با این تفاوت که ۴۵ درجه چرخیده است یعنی یکی از فیلترها در امتداد قطر گوشهٔ پایین سمت چپ به گوشهٔ بالا سمت راست قرار گرفته و دیگری عمود بر آن یعنی در امتداد گوشهٔ پایین سمت راست به گوشه بالا سمت چپ قرار دارد. زوج فیلتر دوم اصطلاحاً **«دستگاه مورب»** (Diagonal Basis) نامیده می‌شود. بدین ترتیب آلیس دارای دو دستگاه مبنا است که می‌تواند به انتخاب خود پرتوی نور را از هر کدام از این دستگاههای مبنا (که هر یک شامل دو فیلتر است) عبور بدهد. البته در دنیای واقعی آلیس دارای چهار فیلتر پلاریزه‌کنندهٔ مجزا نیست بلکه فقط یک قطعه بلور خاص (کریستال پلاریزه کننده) وجود دارد که می‌تواند با سرعت بسیار بالا به هر یک از فیلترهای مورد نظر سوئیچ کند و زاویهٔ پلاریزاسیون را عوض

نماید.[۱] در سمت مقابل، باب نیز از چنین ابزاری استفاده می‌کند. این واقعیت که آلیس و باب هر کدام دارای دو دستگاه مبنا هستند در رمزنگاری کوآنتومی بسیار اساسی و حیاتی است.

آلیس برای هر یک از دستگاههای مبنا یکی از جهتها را بیت صفر و دیگری را بیت یک فرض می‌کند. در مثالی که در ادامه مطرح کرده‌ایم فرض شده که آلیس پلاریزاسیون راستای عمود را بیت صفر و راستای افق را بیت ۱ قرارداد کرده است. همچنین پلاریزاسیون راستای ۴۵ درجه (/) بیت صفر و راستای ۱۳۵ درجه (\ )، بیت ۱ فرض شده است. آلیس قرارداد انتخابی خود را برای باب می‌فرستد.

حال آلیس یک Pad برای خود انتخاب می‌کند که این انتخاب می‌تواند توسط یک مولد اعداد تصادفی انجام شود (موضوعی که به خودی خود مقوله‌ای پیچیده محسوب می‌شود). او این Pad را بیت به بیت برای باب می‌فرستد در حالی که برای ارسال هر بیت بطور تصادفی از یکی از دستگاههای مبنا استفاده می‌کند.[۲]

برای ارسال یک بیت، تفنگ تولید فوتون قادر است فوتونی را منتشر کند که پلاریزاسیون آن کاملاً منطبق با دستگاه مبنای آلیس و به اختیار او باشد. به عنوان مثال او می‌تواند بطور متوالی دستگاه مبنای پلاریزاسیون را تغییر داده و دستگاه مورب (Diagonal)، دستگاه مستقیم (Rectilinear) را انتخاب نماید. مثلاً آلسی می‌تواند برای ارسال یک Pad نظیر 1001110010100110 ، طبق قرارداد فوق فوتونهای پلاریزه شدهٔ شکل ۸–۵–الف را ارسال نماید. با داشتن یک Pad معلوم و مشخص بودن توالی دستگاههای مبنا، پلاریزاسیون مورد استفاده برای هر بیت به صورت یکتا مشخص می‌شود. «بیتهایی که در قالب یک فوتون منفرد و در یک زمان مشخص ارسال می‌شوند اصطلاحاً کوبیت(ها) (Qubit(s)) نامیده می‌شوند.»

	شماره بیت	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	بیتهای داده	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0
(الف)	آنچه آلیس می‌فرستد	\	↕	↕	\	↔	\	/	/	↔	↕	↔	/	/	↔	↔	↕
(ب)	دستگاه مبنای باب	+	+	X	X	X	+	+	X	+	X	+	X	X	X	+	X
(ج)	آنچه باب دریافت می‌کند	↕	↕	/	\	\	↕	↔	/	↔	/	↔	/	/	\	↔	/
(د)	آیا مبناها صحیح بوده‌اند؟	No	Yes	No	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No
(ه)	One-time pad		0		1				0	1		1	0	0		1	
(و)	دستگاه مبنای ترودی	X	+	+	X	+	+	X	+	+	X	X	+	X	+	X	X
(ز)	pad ترودی	x	0	x	1	x	x	x	?	1	x	?	?	0	x	?	x

شکل ۸–۵. مثالی از رمزنگاری کوآنتومی.

۱. به عبارت بهتر یک کریستال به صورت الکترونیکی به گونه‌ای تنظیم می‌شود که رفتار هر یک از فیلترهای مورد نظر را بروز بدهد. امروزه کریستالهای پلاریزه کنندهٔ الکترونیکی نور موجودند. –م.

۲. یعنی در ارسال PAD به صورت «کاملاً تصادفی» برای بیت ۱ یکی از پلاریزاسیونهای / یا ↕ و برای بیت صفر یکی از پلاریزاسیونهای \ یا ↔ انتخاب می‌شود. –م.

باب نمی‌داند که از چه دستگاه مبنا [برای دریافت فوتونها] استفاده کند، لذا برای دریافت هر فوتون یکی از دستگاههای قراردادی خود را به صورت تصادفی در نظر گرفته و آن را بکار می‌گیرد، مثلاً به گونه‌ای که در شکل ۸-۵-ب مشاهده می‌کنید. اگر او تصادفاً برای یک بیت، دستگاه مبنایی مطابق با دستگاه مبنای پلاریزاسیون آلیس انتخاب کرده باشد، بیت صحیح دریافت خواهد کرد ولیکن اگر دستگاه مبنای او اشتباه باشد آنگاه به صورت کاملاً تصادفی و با احتمال مساوی یکی از بیتهای ۱ یا صفر دریافت خواهد شد زیرا طبق نظریهٔ مکانیک کوآنتومی هر گاه یک فوتون به یک فیلتر پلاریزه کننده با اختلاف زاویهٔ ۴۵ درجه (نسبت به زاویهٔ پلاریزاسیون خود فوتون) برخورد کند بطور تصادفی و با احتمال مساوی، یا به زاویه پلاریزاسیون فیلتر و یا به زاویهٔ قائم بر پلاریزاسیون فیلتر، پرش خواهد کرد.این ویژگی فوتونها در مکانیک کوآنتومی یکی از اصول اساسی به شمار می‌آید. بدین ترتیب هر گاه دستگاه مبنای انتخاب شده توسط باب صحیح نباشد برخی از بیتهای دریافتی صحیح و برخی دیگر اشتباه هستند ولیکن باب نمی‌داند که کدام صحیح و کدام غلط هستند. نتیجهٔ دریافت بیتها توسط باب در شکل ۸-۵ نشان داده شده است.

باب چگونه می‌تواند متوجه شود که کدام مبنا درست و کدام غلط بوده است؟ او به سادگی و به صورت آشکار به آلیس اعلام می‌کند که دستگاههای مبنای انتخابی او برای هر بیت چه بوده‌اند و آلیس هم در پاسخ به او خواهد گفت که کدام صحیح و کدام یک غلط است؛ (به گونه‌ای که در شکل ۸-۵-د ملاحظه می‌کنید). با این اطلاعات طرفین می‌توانند یک رشته بیت، مطابق با شکل ۸-۵-ه از حدسهای درست بسازند. بطور میانگین این رشتهٔ بیت، نیمی از کل رشته بیت Pad را تشکیل می‌دهد ولی از آنجایی که طرفین آن را می‌دانند می‌تواند به عنوان کلید رمز (یا همان Pad) انتخاب شود. تمام کاری که آلیس مجبور است انجام بدهد آن است که طول Pad انتخابی در ابتدای کار از دو برابر طول مورد نظر بیشتر باشد تا پس از دریافت، طول رشتهٔ Pad به اندازهٔ مطلوب و مورد باشد. بدین نحو مسئله حل شده است.

اما لحظه‌ای تأمل کنید؛ در این میان ترودی را فراموش کرده‌ایم. فرض کنید او کنجکاو است بداند که آلیس چه می‌کند، لذا فیبرنوری را قطع می‌کند و مدار آشکارساز (Detector) و فرستندهٔ خود را در میانهٔ فیبر جاسازی می‌کند، [و بدین ترتیب یک انشعاب فعال برای استراق سمع پدید می‌آورد]. ترودی نمی‌داند برای هر فوتون حامل داده چه مبنایی را [برای پلاریزاسیون] در نظر بگیرد. بهترین کاری که می‌تواند انجام بدهد آن است که برای هر فوتون یک مبنای تصادفی انتخاب کند، دقیقاً همانند کاری که باب می‌کند. مثالی از انتخابهای تصادفی او در شکل ۸-۵-و نشان داده شده است. بعداً وقتی باب به آلیس (به صورت آشکار) گزارش می‌دهد که مبناهای انتخابی او چه بوده و آلیس نیز به او می‌گوید که کدام صحیح و کدام غلط است، ترودی نیز می‌داند که انتخابهای او کدام صحیح و کدام غلط هستند. مطابق با شکل ۸-۵ او بیتهای ۰ و ۱ و ۲ و ۳ و ۴ و ۶ و ۸ و ۱۲ و ۱۳ را صحیح بدست آورده است ترودی از پاسخ آلیس به باب متوجه می‌شود که فقط بیتهای ۱ و ۳ و ۷ و ۸ و ۱۰ و ۱۱ و ۱۲ و ۱۴ جزو Pad (کلید رمز) هستند. برای چهار تا از این بیتها یعنی (۱۲ و ۸ و ۳ و ۱) او حدس درستی زده است و بیتهای صحیحی بدست آورده است. برای چهار بیت دیگر یعنی (۱۴ و ۱۱ و ۱۰ و ۷) او مبنای صحیحی نداشته است و طبیعتاً نمی‌داند که بیت ارسال شده چه بوده است. بنابراین مطابق با شکل ۸-۵-ه باب می‌داند که Pad او با رشته بیت 01011001 آغاز می‌شود در حالی که آنچه ترودی از این Pad در اختیار دارد مطابق با شکل ۸-۵-ز رشتهٔ 01?1??0? است.

البته آلیس و باب هر دو آگاهند که ممکن است ترودی بخشی از Pad آنها را بدست بیاورد، به همین دلیل مایلند اطلاعاتی که ترودی دارد را کاهش بدهند. لذا آنها می‌توانند با انجام عملیات تبدیل (Transformation) بر روی Pad این کار را انجام بدهند. به عنوان مثال آنها می‌توانند Pad طولانی خود را به بلوکهای ۱۰۲۴ بیتی تقسیم

کرده و سپس آن را به توان دو برسانند تا به عددی ۲۰۴۸ بیتی تبدیل شود؛ سپس بلوکهای ۲۰۴۸ بیتی به هم چسبیده و Pad اصلی (کلید رمز) را تشکیل می‌دهند. در این صورت ترودی با داشتن اطلاعات جزیی از بلوکهای ۱۰۲۴ بیتی [زیرا نیمی از بیتها را به درستی نمی‌داند] قادر نیست توان دوی آن را محاسبه کند و بنابراین چیزی از Pad واقعی نخواهد فهمید. انجام تبدیل بر روی Pad اولیه، (به منظور کاهش اطلاعات ترودی)، اصطلاحاً «**تشدید پنهان‌سازی**»[1] نامیده می‌شود. در دنیای عمل، به جای آن که بلوکهای Pad به توان دو برسد، عملیات پیچیده‌ای بر روی بلوکهای ورودی انجام می‌شود تا هر بیت خروجی به هر بیت ورودی وابستگی داشته باشد. [بدین ترتیب هر گونه حدس غلط در مورد یک بیت، پس از انجام عملیات تبدیل منجر به خطاهای متعدد در Pad نهایی خواهد شد.]

ترودی بی‌نوا! نه تنها هیچ حدسی در مورد Pad در اختیار ندارد بلکه حضور او [و استراق سمع داده‌ها] مهم و محرمانه نیست. گذشته از این، او باید بیتهایی را که از آلیس دریافت می‌کند مجدداً برای باب تقویت و ارسال (رله) نماید تا وانمود کند که باب در حال محاورهٔ مستقیم با آلیس است [تا حضورش کماکان مخفی بماند.] مشکل آن است که بهترین کاری که ترودی می‌تواند انجام بدهد آن است که کیوبیتهای دریافتی (Qubits) را مطابق با پلاریزاسیون فرضی خودش برای باب ارسال کند و چون بطور میانگین نیمی از آنچه ارسال می‌کند [به دلیل اشتباه در دستگاه مبنای پلاریزاسیون] غلط است لذا حجم بیتهای اشتباه در Pad متعلق به باب بسیار زیاد خواهد شد.

وقتی آلیس شروع به ارسال داده‌ها می‌کند آن را با «**کدهای تصحیح خطای پیشرونده**»[2] کدگذاری می‌نماید. از دیدگاه باب یک بیت خطا در Pad معادل با یک بیت خطای انتقال در داده‌ها تلقی می‌شود. به هر حال او به دلیل خطا در Pad یک بیت را به اشتباه دریافت می‌کند. اگر بیتهای تصحیح خطا به همراه داده ارسال شوند او قادر خواهد بود علیرغم وجود خطا، اصل پیام را بازسازی نماید و در عین حال براحتی می‌تواند تعداد خطاهای تصحیح شده را بشمارد. اگر تعداد آنها از حد انتظار بیشتر باشد باب متوجه خواهد شد که ترودی (به عنوان شخص ثالث و مزاحم) در کانال انشعاب ایجاد کرده است و در این حالت می‌تواند طبق دستور عمل نماید؛ (مثلاً به آلیس بگوید که به کانال رادیویی سوئیچ کند یا مستقیماً با پلیس تماس بگیرد و...) البته اگر ترودی قادر باشد یک فوتون را عیناً (مشابه با فوتون دریافتی) تولید نماید می‌تواند فوتونی را دریافت و عین آن را برای باب ارسال کند و بدین ترتیب حضورش کشف نخواهد شد ولیکن هنوز هیچ راهی برای «تولید مثل»[3] فوتونها شناخته نشده است. ولیکن حتی اگر ترودی بتواند فوتونها را تولید مثل نماید از ارزشهای رمزنگاری کوآنتومی برای ایجاد Pad (کلید رمز) کاسته نخواهد شد.

اگرچه نشان داده شده که رمزنگاری کوآنتومی بر روی یک فیبرنوری به طول ۶۰ کیلومتر بخوبی کار می‌کند ولیکن ابزارهای لازم بسیار گران و پیچیده هستند. این نظریه هنوز هم امیدبخش است. برای اطلاعات بیشتر در خصوص رمزنگاری کوآنتومی به کتاب Mullins (۲۰۰۲) مراجعه کنید.

۸-۱-۵ دو اصل اساسی در رمزنگاری

هر چند در صفحاتی که پیش رو دارید چندین سیستم رمزنگاری مختلف را بررسی خواهیم کرد ولیکن فهم دو اصل اساسی که تمام آنها بر آن استوار هستند اهمیت فراوان دارد.

افزونگی (Redundancy)

اولین اصل آن است که تمام پیامهای رمز شده باید شامل مقداری «**افزونگی**» [داده‌های زائد] باشند؛ به عبارت دیگر

---

۱. Privacy Amplification
۲. Forward Error Correcting Code
۳. Cloning

لزومی ندارد که اطلاعات واقعی به همانگونه که هستند رمز و ارسال شوند.[1] یک مثال می‌تواند به فهم دلیل این نیاز کمک کند. فرض کنید یک شرکت به نام TCP[2] (The Couch Potato) با ۶۰۰۰۰ کالا، از طریق سیستم پست الکترونیکی سفارش خرید می‌پذیرد.برنامه‌نویسان شرکت TCP به خیال آن که برنامه‌های مؤثر و کارآمدی می‌نویسند، پیامهای سفارش کالا را مشتمل بر ۱۶ بایت نام مشتری و به دنبال آن سه بایت فیلد داده (شامل یک بایت برای تعداد کالا و دو بایت برای شمارهٔ کالا) در نظر می‌گیرند که سه بایت آخر توسط یک کلید بسیار طولانی رمزنگاری می‌شود و این کلید را فقط مشتری و شرکت TCP می‌داند.

در نگاه اول ممکن است این طرح مطمئن و امن به نظر برسد، از آن جهت که یک اخلالگر غیرفعال (Passive Intruder) به هیچوجه قادر به رمزگشایی اطلاعات نخواهد بود. ولی متأسفانه در این منطق یک اشتباه اساسی وجود دارد که عملاً آن را به طرحی غیرقابل استفاده تبدیل می‌کند. فرض کنید یک کارمند اخراجی کینه‌جو می‌خواهد شرکت TCP را تنبیه کند. لذا قبل از ترک شرکت فهرست مشتریان این شرکت را با خود به همراه می‌برد. [فهرست مشتریان محرمانه و سرّی نیست و رمزنگاری نیز نمی‌شود.] او در طول شب برنامه‌ای می‌نویسد تا با استفاده از نامهای واقعی مشتریان سفارشات جعلی و دروغین بدهد. از آنجایی که او فهرست کلیدهای رمز را در اختیار ندارد لذا در سه بایت آخر مقادیری تصادفی قرار می‌دهد و بدین طریق صدها سفارش جعلی برای شرکت TCP ارسال می‌کند.

وقتی این پیامها دریافت می‌شوند کامپیوتر شرکت TCP ابتدا کلید مربوط به هر مشتری را پیدا می‌کند تا بدنهٔ پیام را رمزگشایی کند. از آنجایی که متأسفانه تمام مقادیر این سه بایت معتبر هستند [یعنی مقادیر تصادفی درون آن پس از رمزگشایی به یک مقدار تصادفی ولی معتبر تبدیل می‌شود] بنابراین کامپیوتر، شروع به چاپ سفارشهای خرید و صورتحساب می‌کند. گرچه شاید عجیب به نظر برسد که یک مشتری ۸۳۷ عدد تاب برای بچه‌ها یا ۵۴۰ جغجغه سفارش بدهد ولی براساس دانشی که یک کامپیوتر دارد شاید خریدار، این مقدار تاب را برای ساخت یک شهربازی در نظر داشته است! بدین ترتیب یک اخلالگر فعال (کارمند اخراجی) توانسته مشکلات بزرگی را برای شرکت TCP ایجاد کند هر چند خود اخلالگر نمی‌تواند بفهمد پیامهایی که کامپیوتر او تولید کرده چه هستند!

این مسئله را می‌توان با اضافه کردن مقداری افزونگی به تمام پیامها حل کرد. به عنوان مثال اگر فیلد سفارش در هر پیام از ۳ بایت به ۱۲ بایت افزایش یابد و نه بایت اول آن باید صفر باشد، آنگاه حملهٔ فوق عملی نخواهد بود زیرا کارمند اخراجی [با تولید اعداد تصادفی ۱۲ بایتی] قادر نخواهد بود یک حجم عظیم از سفارشهای معتبر تولید نماید.[3]

حقیقت قضیه آن است که تمام پیامها باید مقدار قابل توجهی افزونگی (Redundancy) داشته باشند بگونه‌ای که یک اخلالگر فعال (Active Intruder) نتواند پیامهای تصادفی بی‌معنا تولید و ارسال کند و باعث شود این پیامها به عنوان پیامهای معتبر تفسیر شوند.

با این وجود اضافه کردن مقداری «افزونگی» به پیامها باعث می‌شود که رمزشکنها ساده‌تر بتوانند رمز پیامها را بشکنند. فرض کنید که بازار سفارش اجناس از طریق پست الکترونیکی بسیار گرم و رقابتی باشد و رقیب اصلی

---

۱. یعنی یک رشتهٔ رمز شده نباید پس از رمزگشایی معادل با اصل پیام باشد بلکه باید داده‌های زائد و حساب شده‌ای در درون آن جاسازی شده باشد. -م.

۲ اصطلاح The Couch Potato واژه‌ای فکاهی است که معنای کار بیهوده و مسخره دارد و در فارسی در افواه عامه شرکت‌کشک‌سائی ترجمه می‌شود!!!

۳. زیرا آن دسته از اعداد ۱۲ بایتی که پس از رمزگشایی، ۹ بایت اول آنها دقیقاً صفر شود اصلاً مشخص نیستند و تولید آنها به روش سعی و خطا در فضای ۱۲ بایتی ممکن نخواهد بود. -م

شرکت TCP شرکت Sofa Tuber باشد که بشدت علاقمند است بداند شرکت TCP چند جغجغه می‌فروشد! در این راستا از خط تلفن شرکت TCP دزدانه انشعاب گرفته است و اطلاعات خط را استراق سمع می‌کند. در طرح اصلی با سه بایت در بدنهٔ پیام، شکستن رمز و بدست آوردن کلید تقریباً غیرممکن است زیرا پس از حدس زدن یک کلید، رمزشکن هیچ راهی برای اثبات صحت حدس خود ندارد چرا که تقریباً تمام پیامها معتبرند. در طرح جدید با پیامهای ۱۲ بایتی، رمزشکن به سادگی می‌تواند پیامهای معتبر را از پیامهای غیر معتبر تشخیص بدهد. [زیرا از بین حدسهایی که در مورد کلید رمز آزمایش می‌کند حدسی درست است که پیامی با ۹ بایت صفر در ابتدای آن تولید کند.] بهرحال وجود افزونگی الزامی است و داریم:

اصل اساسی ۱ در رمزنگاری : پیامها باید شامل مقداری افزونگی باشند.

به عبارت دیگر پس از رمزگشایی پیام، گیرنده باید بتواند پیامهای معتبر را با یک بررسی و محاسبهٔ ساده [از پیامهایی که به صورت تصادفی تولید شده‌اند] تشخیص بدهد. این افزونگی بدان جهت نیاز است که از ارسال پیامهای بی‌ارزش اخلالگران و فریب خوردن گیرنده در رمزگشایی پیامها و پردازش آنها جلوگیری شود. ولیکن همین افزونگی، شکستن سیستم رمز توسط اخلالگران غیرفعال را ساده‌تر می‌سازد؛ لذا در اینجا یک تناقض پدید می‌آید. بعلاوه افزونگی اطلاعات نباید در قالب اضافه کردن تعداد n تا صفر به ابتدا یا انتهای پیام، انجام بگیرد چراکه اجرای الگوریتمهای رمزنگاری بر روی چنین پیامهایی، نتایج قابل پیش‌بینی برخواهد گرداند و کار رمزشکن را ساده‌تر می‌کند. اضافه کردن یک چندجمله‌ای CRC به جای دنباله‌ای از صفرها بهتر خواهد بود زیرا از یک طرف گیرندهٔ اصلی براحتی می‌تواند صحت پیامها را بررسی کند و از طرف دیگر رمزشکن را با حجم کار بسیار زیاد مواجه می‌کند [تا عمل رمزشکنی ناموفق باشد]. حتی بهتر از آن استفاده از «توابع درهم‌سازی رمز» (Hash) است، مفهومی که بعداً آن را بررسی خواهیم کرد.

اگر برای لحظاتی به رمزنگاری کوآنتومی برگردیم، می‌توانیم به نقشی که «افزونگی» در آن سیستم ایفاء می‌کند پی ببریم. به دلیل استراق سمع فوتونها توسط ترودی، برخی از بیتها در Pad انتخابی باب، غلط خواهد بود. بنابراین باب نیازمند به افزونگی در پیامهای دریافتی خود است تا بتواند خطاهای موجود در آن را تشخیص بدهد. یک روش خام و بسیار ضعیف برای ایجاد افزونگی در پیام آن است که پیام دو بار تکرار شود. اگر دو نسخهٔ تکراری پیام مشابه نباشند، باب متوجه خواهد شد که یا فیبرنوری بشدت نویزی است و یا آن که شخصی در حال استراق سمع از روی کانال انتقال است. البته دو بار ارسال کردن یک پیام واحد بسیار غیرمنطقی و بیهوده است لذا اضافه کردن «کدهای همینگ» (Hamming Code) یا کدهای «رید سولومون» (Reed-Solomon Code) راه مؤثر و کارآمدتری برای کشف و تصحیح خطا محسوب می‌شود. ولی بهرحال روشن است که به منظور تشخیص پیامهای معتبر از پیامهای ساختگی به مقداری افزونگی نیاز خواهد بود، مخصوصاً وقتی پای یک اخلالگر فعال (Active Intruder) در میان است.

تازگی پیامها (Freshness)

دومین اصل اساسی در رمزنگاری آن است که باید محاسباتی صورت بگیرد تا مطمئن شویم هر پیام دریافتی تازه و جدید است یا به عبارتی اخیراً فرستاده شده است. این بررسی برای جلوگیری از ارسال مجدّد پیامهای قدیمی توسط یک اخلالگر فعال، الزامی است. اگر چنین بررسیهایی انجام نشود کارمند اخراجی ما [در نمایشنامهٔ قبلی] قادر است با ایجاد یک انشعاب مخفی از خط تلفن، پیامهای معتبری را که قبلاً ارسال شده، مکرراً ارسال نماید. [حتی اگر نداند محتوای آن چیست.] این نظریه را می‌توان در قالب زیر بازگو کرد:

اصل اساسی ۲ در رمزنگاری : به روشهایی نیاز است تا از حملات منجر به تکرار پیام جلوگیری شود.

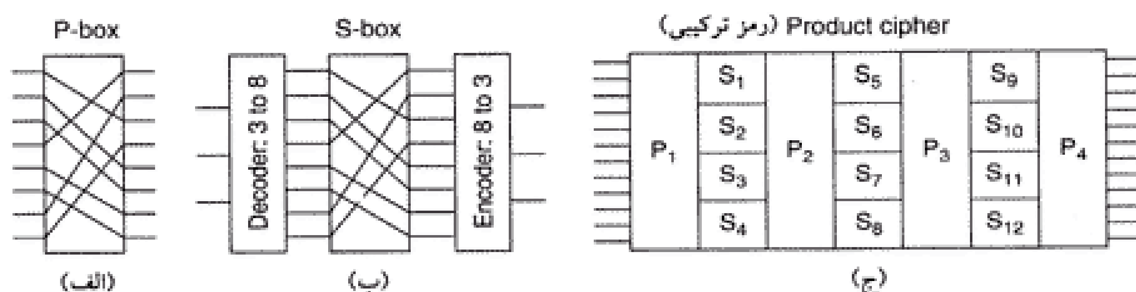
یک چنین محاسبه‌ای را می‌توان با قرار دادن یک «مُهر زمان» (Timestamp) در پیامها پیش‌بینی کرد به نحوی که پیام مثلاً برای ده ثانیه معتبر باشد. گیرندهٔ پیام می‌تواند آن را برای حدود ده ثانیه نگه دارد تا بتواند پیامهای جدید را با آن مقایسه کرده و نسخه‌های تکراری را حذف نماید. پیامهایی که بعد از ده ثانیه دریافت شوند کنار گذاشته می‌شوند؛ بدین ترتیب پیامهای تکراری که دارای مُهر زمان هستند، به عنوان پیامهای قدیمی شناخته و حذف خواهند شد. به غیر از روش مُهر زمان (Timestamp)، روشهای دیگری برای ارزیابی تازگی پیام وجود دارد که در ادامه تشریح خواهند شد.

۸-۲ الگوریتمهای رمزنگاری با کلید متقارن (Symmetric-Key)

روشهای پیشرفته و پیچیدهٔ رمزنگاری از اصول و قواعدی مشابه با رمزنگاری سنتی (مثل روشهای جانشینی و جایگشتی) بهره گرفته‌اند درحالیکه که راهکارها متفاوت هستند. در قدیم رمزنگاران از الگوریتمهای ساده‌ای استفاده می‌کردند در حالی که امروزه عکس این موضوع صادق است: هدف آن است که یک الگوریتم به قدری پیچیده و بغرنج طراحی شود که حتی اگر رمزشکن تودهٔ عظیمی از متن رمز شده را به انتخاب خود در اختیار بگیرد، بدون کلید هرگز نتواند چیزی از بطن آن استخراج کند.

اولین گروه الگوریتمهای پیشرفتهٔ رمزنگاری که در درس امروز به بررسی آنها خواهیم پرداخت «**الگوریتمهای با کلید متقارن**» نام دارد زیرا این الگوریتمها چه برای رمزنگاری و چه برای رمزگشایی از یک کلید مشابه استفاده می‌کنند. شکل ۸-۲ عملکرد یک الگوریتم با کلید متقارن را به تصویر کشیده است. بطور خاص بر روی رمزهای بلوکی متمرکز خواهیم شد که در آنها یک بلوک n بیتی از «متن آشکار» تحویل الگوریتم شده و براساس کلید تبدیلاتی بر روی آن انجام و یک بلوک n بیتی «رمز» بدست می‌آید.

الگوریتمهای رمزنگاری را می‌توان هم به صورت سخت‌افزاری (به منظور سرعت بالاتر) و هم به صورت نرم‌افزاری (برای انعطاف‌پذیری بیشتر) پیاده‌سازی کرد. اگرچه توجه ما بیشتر معطوف به الگوریتمها و پروتکلهای رمزنگاری، مستقل از پیاده‌سازی واقعی آنهاست ولی توضیحی کوتاه در خصوص سخت‌افزار رمزنگاری بعضاً جالب و قابل توجه خواهد بود. روشهای جانشینی و جایگشتی می‌توانند با یک مدار سادهٔ الکترونیکی پیاده‌سازی شوند. شکل ۸-۶-الف ابزاری را نشان می‌دهد که به نام P-box (P مخفف Permutation یا جایگشت) مشهور است و برای جایگشت بیتهای یک ورودی هشت بیتی کاربرد دارد. در این ساختار اگر بیتهای ورودی را از بالا به پایین با شماره‌های 01234567 شماره‌گذاری کنیم، خروجی این P-box خاصّ، 36071245 خواهد بود. با سیم‌بندی و برنامه‌ریزی درونی، این P-box قادر است هر گونه جایگشت بیتی را عملاً با سرعتی نزدیک به سرعت نور انجام بدهد؛ چرا که هیچگونه محاسبه‌ای لازم نیست و فقط تأخیر انتشار سیگنال وجود دارد. این طراحی از اصل کرکهف تبعیت می‌کند یعنی: حمله‌کننده از روش عمومی جایگشت بیتها مطلع است. آنچه که او از آن خبر ندارد آن است که کدام بیت به کدام بیت نگاشته می‌شود؛ کلید رمز همین است.



شکل ۸-۶. عناصر پایه در رمزنگاری ترکیبی. (الف) P-Box (ب) S-Box (ج) ترکیب این دو.

عمل جانشینی به گونه‌ای که در شکل ۸-۶-ب نشان داده شده توسط بلوکی به نام S-box انجام می‌شود. در این مثال یک ورودی سه بیتی به بلوک S-box وارد شده و یک رمز سه بیتی از آن خارج می‌شود. ورودی سه بیتی، یکی از هشت خط خروجی بخش اول را فعال کرده و آن را به ۱ تنظیم می‌کند در حالی که بقیهٔ خطوط صفر هستند. بخش دوم یک P-box است. سومین بخش، خط انتخاب شدهٔ ورودی را مجدداً در سه بیت کُد می‌کند. منطبق با سیم‌بندی مثال ۸-۶-ب اگر هشت عدد اکتال (مبنای هشت) ورودی را به ترتیب ۰۱۲۳۴۵۶۷ در نظر بگیریم، توالی خروجی به ترتیب عبارتند از ۲۴۵۰۶۷۱۳ . به عبارت دیگر کد صفر با ۲ جابجا می‌شود، ۱ با ۴ جابجا می‌شود و به همین ترتیب. در این ساختار نیز با سیم‌بندی مناسب در بخش P-box از S-box ، هر گونه جانشینی دلخواه ممکن و میسر خواهد بود. به علاوه چنین ابزاری را می‌توان با سخت‌افزار پیاده کرد و سرعت بسیار بالایی را بدست آورد زیرا بلوکهای کدکننده (Encoder) و دیکودکننده (Decoder) فقط دارای یک یا دو گیت با تأخیر بسیار ناچیز (کسری از نانوثانیه) است و تأخیر انتشار بخش P-box می‌تواند کمتر از یک پیکو ثانیه باشد.

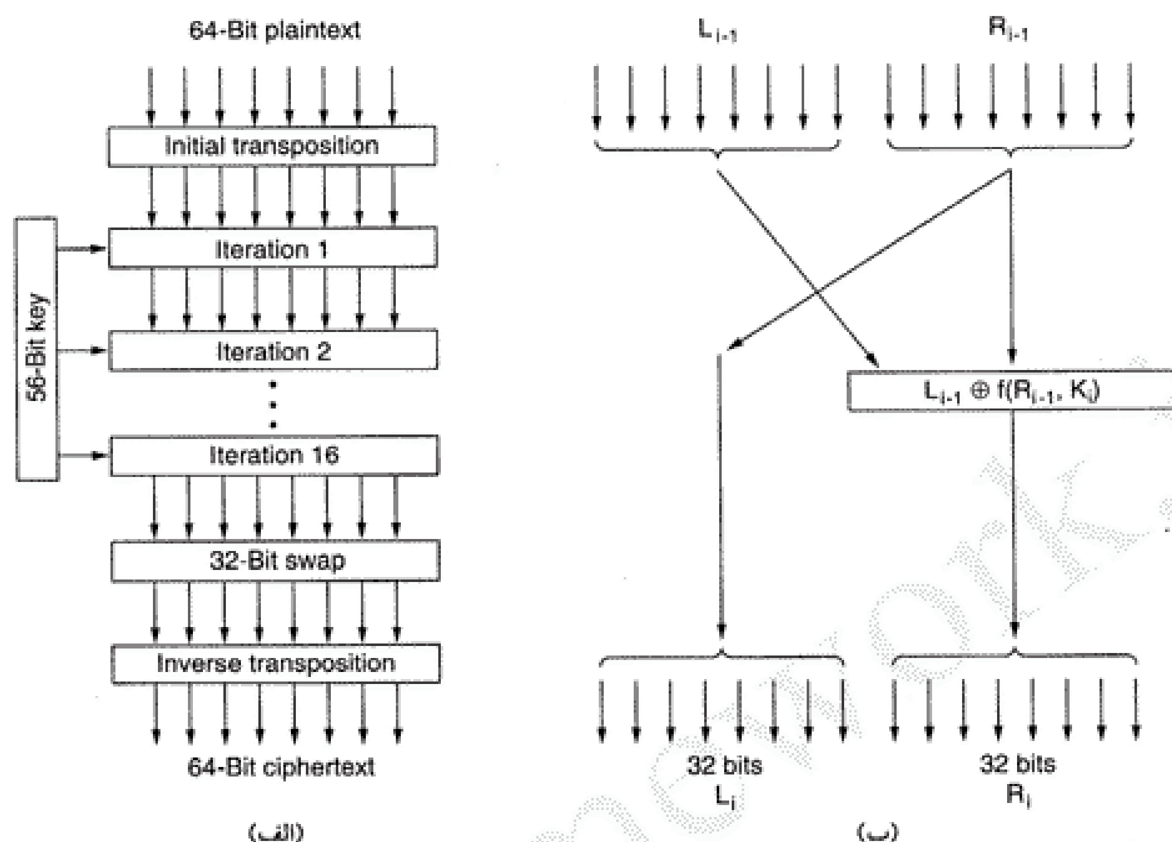
قدرت واقعی این عناصر پایه ، زمانی مشخص می‌شود که بخواهیم با ترکیب تعدادی از این بلوکها همانند شکل ۸-۶-ج، یک سیستم رمزنگار ترکیبی ایجاد نماییم. در این مثال در مرحلهٔ اول، به ۱۲ خط ورودی جایگشت داده می‌شود (P_1). در تئوری می‌توان بلوکی داشت که مستقیماً یک عدد ۱۲ بیتی را به عدد ۱۲ بیتی دیگر بنگارد ولی در عمل چنین ابزاری به $۲^{۱۲}$ یعنی ۴۰۹۶ اتصال متقاطع (در بخش دوم) نیاز خواهد داشت. بجای آن، ورودیها به چهار دستهٔ سه بیتی شکسته شده و هر یک از این سه بیت به صورت مستقل جایگشت داده می‌شود. اگرچه این روش کمتر معمول است ولیکن در نوع خود قدرتمند بشمار می‌رود. با قرار دادن تعداد زیاد و کافی از این بلوکها در ساختار ترکیبی فوق، می‌توان سیستمی را پیاده کرد که خروجی آن (به عنوان تابعی از ورودی) به قدر کافی پیچیده و بهم ریخته باشد.

امروزه سیستمهای رمزنگار ترکیبی که بر روی k بیت ورودی عمل کرده و k بیت خروجی تولید می‌کنند بسیار رایج هستند. بطور معمول k بین ۱۶ تا ۲۵۶ متغیر است. یک سخت افزار رمزنگار ترکیبی برخلاف شکل ۸-۶-ج به جای هفت بخش حداقل هجده بخش فیزیکی متوالی دارد. پیاده‌سازی نرم‌افزاری این روش، در قالب یک حلقه با حداقل هشت تکرار، برنامه‌نویسی می‌شود و در هر تکرار عملکرد یکی از S-boxها بر روی یک بلوک ۶۴ تا ۲۵۶ بیتی انجام و سپس عملیات جایگشت محاسبه می‌شود. اغلب در شروع، یک جایگشت ابتدایی بر روی داده‌ها انجام می‌شود سپس عملیات رمزنگاری انجام شده و در پایان نیز یک مرحله جایگشت تکمیلی انجام می‌گیرد. در ادبیات رمزنگاری هر مرحلهٔ تکرار یک «دور» (Round) نامیده می‌شود.

۸-۲-۱ رمزنگاری DES (Data Encryption Standard)

در ژانویه ۱۹۷۷، دولت ایالات متحده یک سیستم رمزنگاری ترکیبی را که توسط IBM طراحی شده بود به عنوان استاندارد رمزنگاری اطلاعات و اسناد رسمی و طبقه‌بندی نشدهٔ خود پذیرفت. از این رمز یعنی DES ، بطور گسترده‌ای توسط صنایع در محصولات امنیتی استفاده شد. البته این سیستم با شکل اولیه و اصلی آن چندان امن نیست ولی شکل اصلاح شدهٔ آن هنوز هم سودمند است. در این مبحث چگونگی عملکرد DES را تشریح خواهیم کرد.

طرح کلی سیستم DES در شکل ۸-۷-الف نشان داده شده است. متن آشکار در قالب بلوکهای ۶۴ بیتی (۸ کاراکتری) رمزنگاری می‌شود و نهایتاً متن ۶۴ بیتی رمز در خروجی بدست می‌آید. این الگوریتم رمزنگاری که دارای یک پارامتر ۵۶ بیتی به عنوان کلید است، عملیات لازم را در ۱۹ مرحلهٔ مجزا انجام می‌دهد. اولین مرحله، شامل یک جایگشت بر روی متن ۶۴ بیتی ورودی است که این جایگشت مستقل از کلید رمز است [یعنی کلید در این مرحله وارد نخواهد شد و جدول جایگشت ثابت است.] آخرین مرحله (مرحلهٔ نوزدهم) دقیقاً عکس این



شکل ۸-۷. استاندارد DES (الف) الگوی کلی (ب) جزئیات یکی از مراحل تکرار. علامت ⊕ بمعنای XOR است.

جایگشت انجام می‌گیرد. مرحله قبل از آخر نیز، جابجایی ۳۲ بیت سمت چپ با ۳۲ بیت سمت راست می‌باشد. ۱۶ مرحلهٔ باقیمانده از لحاظ عملکرد دقیقاً مشابهند، با این تفاوت که در هر مرحله، از پارامتری متفاوت که براساس کلید تعیین می‌گردد، استفاده شده است. الگوریتم به گونه‌ای طراحی شده که اجازه می‌دهد رمزگشایی اطلاعات توسط همان کلیدی انجام شود که رمزنگاری نیز توسط آن انجام شده بود؛ خصوصیتی که در تمام الگوریتمهای با کلید متقارن مورد نیاز است. مراحل رمزگشایی اطلاعات دقیقاً برعکس مراحل رمزنگاری است.

عملکرد یکی از مراحل میانی، در شکل ۸-۷-ب نشان داده شده است. در هر مرحله، دو ورودی ۳۲ بیتی وارد و دو رشتهٔ ۳۲ بیتی خروجی، تولید می‌شود. رشتهٔ ۳۲ بیتی سمت چپ در خروجی دقیقاً مشابه رشتهٔ ۳۲ بیتی سمت راست است. رشتهٔ ۳۲ بیتی سمت راست حاصل عمل XOR رشته ورودی سمت چپ با تابعی از رشتهٔ سمت راست و کلید این مرحله یعنی K_i است. تمام پیچیدگی روش، در این تابع نهفته است.

تابع f شامل چهار گام متوالی است: ابتدا با توسعهٔ عدد ۳۲ بیتی R_{i-1} (رشتهٔ ۳۲ بیتی سمت راست ورودی) یک عدد ۴۸ بیتی به نام E بدست می‌آید که براساس یک جایگشت ساده و تکرار برخی از بیتها انجام می‌شود.[1] در مرحلهٔ دوم E و K_i با یکدیگر XOR می‌شوند. [E حاصل مرحلهٔ قبل و K_i کلید این مرحله است که از کلید ۵۶ بیتی اصلی بدست می‌آید.] خروجی این مرحله به هشت گروه شش بیتی تقسیم شده و هر یک از آنها به یک S-box خاص وارد می‌شوند. هر یک از ۶۴ حالت ممکن ورودی، توسط S-box به یک خروجی چهار بیتی نگاشته می‌شود. نهایتاً هشت خروجی چهار بیتی (جمعاً ۳۲ بیت) از درون یک P-box (بمنظور جایگشت بیتی) گذر داده می‌شود.

۱. به ازای هر ۴ بیت، دو بیت تکراری (به ابتدا و انتهای آن چهار بیت) اضافه می‌شود. رجوع کنید به کتاب استالینگ. -م.

در هر یک از ۱۶ مرحله، الگوریتم فوق ثابت است ولی کلید متفاوتی بکار گرفته می‌شود. قبل از آن که الگوریتم آغاز شود، یک جایگشت ۵۶ بیتی بر روی کلید اعمال می‌شود [تا بعداً از این کلید ۵۶ بیتی، شانزده کلید رمز ۴۸ بیتی برای هر مرحله بدست آید.] دقیقاً قبل از شروع هر مرحله، کلید به دو بخش ۲۸ بیتی تقسیم شده و هر یک از این بخشها برحسب شمارهٔ مرحله، به سمت چپ شیفت گردشی (Rotation) داده می‌شوند. [مثلاً برای کلید مرحلهٔ پنجم هر بخش ۲۸ بیتی، پنج بیت به سمت چپ شیفت گردشی داده می‌شود.] پس از این مرحله برای محاسبهٔ هر کلید یعنی K_i یک جایگشت ۵۶ بیتی دیگر بر روی آن اعمال شده و نهایتاً یک زیرمجموعهٔ چهل و هشت بیتی از این ۵۶ بیت استخراج و در هر مرحله از آن استفاده می‌شود.

تکنیکی که برخی از اوقات برای قدرتمندتر کردن سیستم DES بکار می‌رود اصطلاحاً **«سفیدسازی»** (Whitening) نام گرفته است. برای این کار هر بلوک ۶۴ بیتی ورودی به سیستم DES، ابتدا با یک کلید ۶۴ بیتی تصادفی XOR می‌شود؛ سپس خروجی DES مجدداً با کلید دوم XOR می‌گردد. عمل «سفیدسازی» به سادگی برگشت‌پذیر است و برای این کار عکس عملیات فوق انجام می‌شود (به شرط آن که کلید سفیدسازی در اختیار گیرنده باشد). از آنجایی که این تکنیک از دو کلید ۶۴ بیتی اضافی استفاده می‌کند، طول کلیدها افزایش خواهد یافت، لذا فضای جستجوی کلید (به روش سعی و خطا) را افزایش داده و رمزشکنی آن را بسیار وقتگیر خواهد کرد. دقت کنید که برای تمام بلوکهای ۶۴ بیتی از یک کلید سفیدسازی مشترک استفاده می‌شود. (به عبارت بهتر تنها یک کلید سفیدسازی وجود دارد.)

از زمانی که DES معرفی و بکار گرفته شد بحث و مناقشات گسترده‌ای پیرامون آن در گرفت. این سیستم مبتنی بر یک روش رمزنگاری به نام **«لوسیفر»** (Lucifer) است که IBM آن را طراحی و ثبت کرده بود با این تفاوت که IBM در آن از کلید رمز ۱۲۸ بیتی به جای ۵۶ بیتی استفاده می‌کرد. وقتی دولت فدرال ایالات متحده خواست که یک سیستم رمزنگاری را برای پرونده‌های طبقه‌بندی نشده استاندارد کند، IBM را دعوت کرد تا روش خود را برای NSA، تشریح کند. (آژانس امنیت ملی یا NSA در دولت ایالات متحده، بزرگترین مجموعهٔ ریاضیدانان و رمزشکنهای دنیاست). NSA گروهی به شدت سرّی است تا جایی که در مورد آن یک لطیفه وجود دارد:

س. NSA مخفف چیست؟

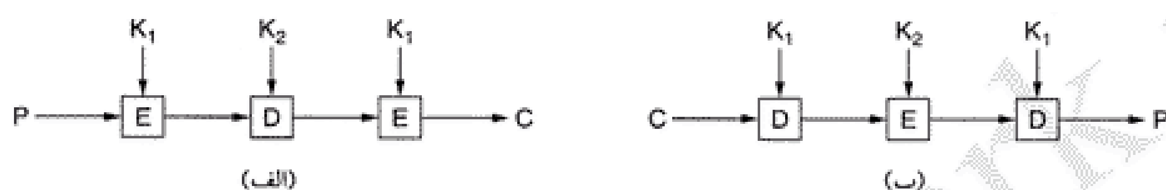
ج. مخفف No Such Agency به معنای «اصلاً چنین آژانسی وجود ندارد!» اما در واقع NSA مخفف کلمات National Security Agency (آژانس امنیت ملی) است.

پس از تشریح روش، IBM کلید ۱۲۸ بیتی را به ۵۶ بیت کاهش داد و تصمیم گرفت فرآیند طراحی DES را محرمانه نگه دارد. بسیاری از افراد شک کردند که شاید دلیل کاهش طول کلید آن بوده که NSA بتواند در صورت لزوم رمز DES را بشکند در حالی که هیچ سازمانی با بودجهٔ کمتر قادر به چنین کاری نباشد. از طرفی محرمانه بودن طرح، احتمالاً بدان دلیل بوده که در آن یک رخنهٔ عمدی گذاشته شده تا NSA راحتتر بتواند رمز DES را بشکند. وقتی یکی از ماموران NSA به صورت محترمانه و غیررسمی از IEEE خواست که برنامهٔ کنفرانس رمزنگاری خود را لغو نماید تا کار عموم را برای رمزشکنی راحتتر از قبل نکند، دامنهٔ این شایعات قوت گرفت. بعداً NSA همه چیز را انکار کرد!

در سال ۱۹۷۷ دو محقق رمزنگاری از دانشگاه استنفورد به نامهای «دیفی» و «هلمن» ماشینی برای شکستن رمز DES طراحی کردند و تخمین زدند که این ماشین با هزینه‌ای حدود بیست میلیون دلار قابل ساخت است. این ماشین می‌توانست با داشتن یک قطعهٔ کوچک از متن آشکار و متن رمز شده، کلید رمز را از بین 2^{56} حالت مختلف، در زمانی کمتر از یک روز پیدا کند. امروزه چنین ماشینی زیر یک میلیون دلار قیمت دارد.

Triple DES (رمزنگاری سه گانه)

در همان اَوان ۱۹۷۹، IBM به این حقیقت پی برده بود که طول کلید در سیستم DES بیش از اندازه کوتاه است و روشی برای افزایش مؤثر طول کلید با استفاده از «**رمزنگاری سه‌گانه**» ابداع کرد. (Tuchman 1979) روش انتخابی آنها که بعداً در قالب استاندارد بین‌المللی ۸۷۳۲ (IS 8732) معرفی شد، در شکل ۸-۸ نشان داده شده است. در این شکل از دو کلید و سه مرحله پردازش استفاده شده است. در مرحلهٔ اول، متن اصلی با کلید K_1 و مبتنی بر روش معمولی DES رمزنگاری می‌شود. در مرحلهٔ بعدی DES در حالت **رمزگشایی** ولی با کلید K_2 بر روی نتیجهٔ مرحلهٔ قبل اعمال می‌شود. نهایتاً بار دیگر رمزنگاری DES با کلید K_1 (کلید اول) انجام می‌شود.



شکل ۸-۸ (الف) سه بار رمزنگاری بکمک استاندارد DES. (ب) رمزگشایی.

این سبک طراحی بلافاصله دو سؤال را به پیش می‌کشد. اول آن که چرا به جای سه کلید فقط از دو کلید رمز استفاده شده است؟ ثانیاً چرا به جای سه بار رمزنگاری متوالی (اصطلاحاً *EEE*[1]) از روش «رمزنگاری-رمزگشایی-رمزنگاری» (اصطلاحاً EDE[2]) استفاده شده است؟

دلیل آن که فقط از دو کلید استفاده شده است آن است که حتی وسواسی‌ترین رمزنگاران اذعان دارند که برای کاربردهای تجاری کنونی یک کلید ۱۱۲ بیتی بخوبی کفایت می‌کند و مطمئن است. (در بین رمزنگاران، وسواس و تردید یک ویژگی ممتاز تلقی می‌شود نه یک اشکال!) حرکت به سمت کلید ۱۶۸ بیتی [۳کلید ۵۶بیتی] در مقایسه با بهرهٔ واقعی آن، مشکلات سربار زیادتری در خصوص مدیریت و حمل و نقل کلید اضافی ایجاد خواهد کرد.

دلیل استفاده از روش «رمزنگاری-رمزگشایی-رمزنگاری» (EDE) سازگار ماندن آن با سیستم تک کلیدی DES بوده است. هر دو عمل رمزنگاری یا رمزگشایی در حقیقت نگاشت یک عددِ ۶۴ بیتی به عددِ ۶۴ بیتی دیگر هستند. استفاده از روش EDE به جای EEE این امتیاز بزرگ را دارد که یک کامپیوتر که مبتنی بر «DES سه‌گانه» عمل می‌کند، براحتی قادر است با انتخاب $K_1=K_2$ این سیستم را به DES معمولی تبدیل کرده و با ماشینهایی که سیستم رمزنگاری آنها DES تک کلیدی است محاوره داشته باشد. در آن زمان، این ویژگی اجازه می‌داد که سیستم «DES سه‌گانه» به تدریج در محیطها جا بیفتد و با سیستم‌های رمزنگاری قدیمی سازگار باشد؛ این خصوصیت اگرچه برای رمزنگاران دانشگاهی اهمیتی ندارد ولی برای شرکت IBM و مشتریان آن بسیار حیاتی بود!

۸-۲-۲ استاندارد پیشرفتهٔ رمزنگاری: AES

در حالی که DES آرام آرام به پایان عمر خود نزدیک می‌شد (حتی با ابداع DES سه گانه)، برای NIST[3] در ایالات متحده (که مسئولیت بهبود استانداردهای دولت فدرال آمریکا را برعهده گرفته است)، مسجّل شد که دولت به یک استاندارد رمزنگاری جدید برای اسناد طبقه‌بندی نشده خود نیاز مبرم دارد. NIST به فراست از دشمنان DES آگاه بود و نیک می‌دانست که اگر به یکباره استاندارد جدیدی را معرفی نماید همهٔ آنانی که دستی در رمزنگاری دارند ناخودآگاه فرض را بر آن می‌گذارند که باز هم آژانس سرویسهای محرمانهٔ ایالات متحده، (NSA) یک **رخنه**[4] در

---

۱. Encrypt-Encrypt-Encrypt
۲. Encrypt-Decrypt-Encrypt
۳. National Institute of Standard Technology
۴. در پشتی یا Backdoor

این سیستم باقی گذاشته است و هر چیزی که با آن رمز شود توسط NSA رمزگشایی و خوانده خواهد شد. در این شرایط هیچکس از استاندارد جدید استفاده نمی‌کرد و به احتمال زیاد استاندارد جدید نیز در یک مرگ آرام رو به زوال می‌رفت!

بدین ترتیب NIST در فضای بوروکراسی دولتی، راهکار بسیار جالب و متفاوتی را اتخاذ کرد: او از یک رقابت انتخاباتی در بین رمزنگاران بهره گرفت. در ژانویه ۱۹۹۷ از تمام محققین رمزنگاری دنیا دعوت شد که پیشنهادات خود را برای تدوین یک استاندارد جدید که AES نامگذاری شده بود (استاندارد پیشرفتهٔ رمزنگاری) ارسال نمایند. شرایط شرکت در این رقابت عبارت بودند از:

۱. الگوریتم پیشنهادی باید یک سیستم رمز متقارن و بلوکی باشد.
۲. جزئیات طراحی باید مشخص و عمومی باشد.
۳. باید از کلیدهای ۱۲۸، ۱۹۲ و ۲۵۶ بیتی حمایت شود.
۴. پیاده‌سازی سخت‌افزاری و نرم‌افزاری الگوریتم ممکن باشد.
۵. الگوریتم باید عمومی (غیرانحصاری) یا تحت قوانین غیرانحصاری ثبت شده باشد.[1]

پانزده طرح پیشنهادی قابل توجه ارائه گردید و در همین راستا یک کنفرانس همگانی ترتیب داده شد تا در آن طرحها ارائه شود و شرکت کنندگان تشویق شوند تا اشکالات آنها را یافته و تحلیل کنند. در آگوست ۱۹۹۸، NIST براساس ویژگیهای «امنیت»، «کارآیی»، «سادگی»، «قابلیت انعطاف» و «فضای حافظهٔ مورد نیاز برای پیاده‌سازی» (که در سیستمهای درونکار ـ Embedded ـ بسیار مهم است) پنج طرح برگزیده را انتخاب و معرفی کرد. کنفرانسهای زیادی برگزار شد و هر کسی تیری در تاریکی انداخته بود! عاقبت در کنفرانس نهایی یک رای‌گیری آزادانه انجام شد. برگزیدگان نهایی و امتیازات آنها به ترتیب زیر بود:

۱. Rijndael (توسط John Daemen و Vincent Rijmen) ۸۶ رأی
۲. Serpent (توسط Ross Anderson، Eli Biham و Lars Knudsen) ۵۹ رأی
۳. Twofish (توسط تیمی به سرپرستی Bruce Schneier) ۳۱ رأی
۴. RC6 (توسط آزمایشگاه RSA) ۲۳ رأی
۵. MARS (توسط IBM) ۱۳ رأی

در اکتبر ۲۰۰۰، NIST اعلام کرد که او هم به روش Rijndael رأی می‌دهد و در نوامبر ۲۰۰۱ روش Rijndael استاندارد دولت ایالات متحده شد و در سند استاندارد FIPS 197 ثبت گردید. به دلیل فضای آزاد شگفت‌انگیز حاکم بر این رقابت و همچنین ویژگیهای فنی برتر Rijndael و با توجه بدان که تیم پیشنهاددهنده دو رمزنگار جوان بلژیکی بودند (که شائبهٔ وجود رخنهٔ مورد نظر NSA در آن را از اذهان می‌زداید)، انتظار می‌رود که Rijndael لااقل برای یک دهه استاندارد رمزنگاری کل دنیا شود. روش Rijndael کم و بیش به صورت «راین دال» تلفظ می‌شود و از نام خانوادگی ابداع‌کنندگان آن (Rijmen & Daemon) گرفته شده است.

Rijndael از کلید و بلوکهای دادهٔ ۱۲۸ یا ۲۵۶ بیتی (در قطعات ۳۲ بیتی) حمایت می‌کند؛ طول کلید و طول بلوکهای داده می‌تواند مستقل از هم انتخاب شود. با این وجود استاندارد AES بیان می‌کند که اندازهٔ بلوک داده باید صرفاً ۱۲۸ بیتی باشد ولی طول کلید می‌تواند یکی از سه حالت ۱۲۸، ۱۹۲ و ۲۵۶ انتخاب شود. برای کسی که همواره از کلیدهای ۱۹۲ بیتی استفاده می‌کند استفاده از دو گزینهٔ دیگر AES یعنی یک کلید ۱۲۸ بیتی با بلوک دادهٔ ۱۲۸ بیتی و کلید ۲۵۶ بیتی با بلوک دادهٔ ۱۲۸ بیتی، او را [در خصوص استفاده از آنها] به تردید خواهد افکند.

---
۱. یعنی امتیاز آن متعلق به هیچ کس نباشد.

در بخشی که در ادامه، پیرامون این الگوریتم خواهیم داشت فقط گزینهٔ ۱۲۸/۱۲۸ (کلید رمز ۱۲۸ بیتی / بلوک دادهٔ ۱۲۸ بیتی) را بررسی کرده‌ایم زیرا احتمالاً در کاربردهای معمول دنیای اقتصاد، کلید ۱۲۸ بیتی جا خواهد افتاد. کلید ۱۲۸ بیتی یک فضای حالت با 2^{128} (3×10^{38}=) حالت مختلف ایجاد می‌کند. حتی اگر NSA سفارش ساخت ماشینی با یک میلیارد پردازندهٔ موازی بدهد و هر یک از پردازنده‌ها بتوانند یک کلید را در یک پیکوثانیه (10^{-12}Sec) آزمایش کنند، آزمایش تمام این کلیدها حدود 10^{10} سال طول خواهد کشید. در آن زمان خورشید خاموش گشته است و مردم مجبورند نتیجهٔ کار را در زیر نور شمع بخوانند!!

Rijndael

از دیدگاه ریاضی، رمزنگاری Rijndael مبتنی بر «نظریهٔ میدان گالوا» است که به آن ویژگیهای امنیتی قابل اثبات و مطمئنی بخشیده است. با این وجود می‌توان آن را با کد زبان C بررسی کرد بدون آن که وارد جزئیات ریاضی آن شد.

همانند DES، رمزنگار Rijndael نیز از روشهای جانشینی (Substitution) و جایگشتی (Permutation) استفاده کرده است. همچنین کل مراحل از چندین «دوره» (Round) تشکیل شده است. تعداد «دور» بستگی به طول کلید و اندازهٔ بلوک داده خواهد داشت: از ۱۰ دور برای کلید ۱۲۸ بیتی با بلوک دادهٔ ۱۲۸ بیتی تا ۱۴ دور برای بزرگترین کلید [۲۵۶ بیتی] و بزرگترین بلوک داده [۲۵۶ بیتی].[۱] ولی برخلاف DES، عملیات بر روی بایتها انجام می‌گیرد نه بیتها؛ بدین ترتیب پیاده‌سازی سخت افزاری یا نرم‌افزاری آن ساده‌تر و کارآمدتر خواهد بود. کلیات کُد این الگوریتم در شکل ۸-۹ آورده شده است.

```
#define LENGTH 16                        /* # bytes in data block or key */
#define NROWS 4                          /* number of rows in state */
#define NCOLS 4                          /* number of columns in state */
#define ROUNDS 10                        /* number of iterations */
typedef unsigned char byte;              /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                 /* loop index */
  byte state[NROWS][NCOLS];              /* current state */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];       /* round keys */

  expand_key(key, rk);                   /* construct the round keys */
  copy_plaintext_to_state(state, plaintext);  /* init current state */
  xor_roundkey_into_state(state, rk[0]);      /* XOR key into state */

  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                 /* apply S-box to each byte */
      rotate_rows(state);                /* rotate row i by i bytes */
      if (r < ROUNDS) mix_columns(state);    /* mix function */
      xor_roundkey_into_state(state, rk[r]); /* XOR key into state */
  }
  copy_state_to_ciphertext(ciphertext, state);          /* return result */
}
```

شکل ۸-۹. الگوی کلی برنامهٔ Rijndael.

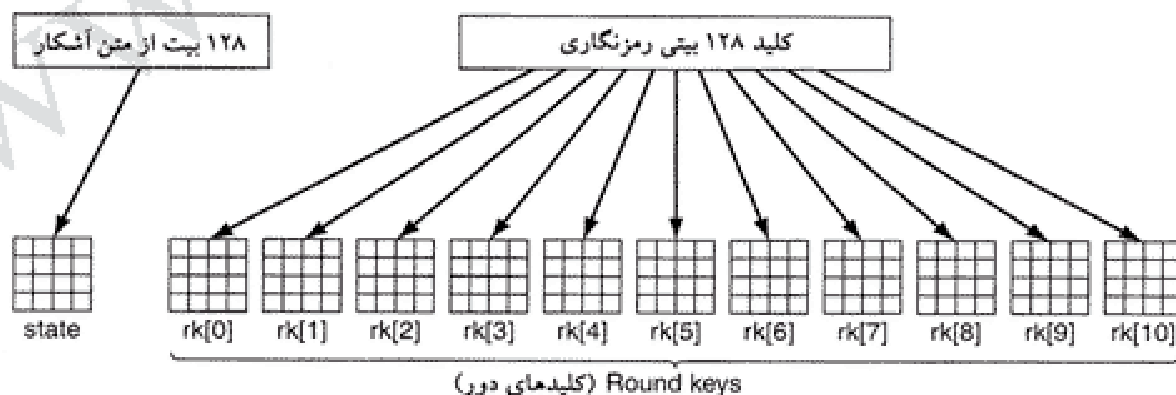
۱. دقت کنید که در رمزنگاری راینْدال طول بلوکهای داده می‌تواند ۲۵۶ بیتی نیز باشد و آنکه در بخش قبلی در مورد AES عنوان شد (که طول بلوک داده باید «صرفاً ۱۲۸ بیتی» باشد) استاندارد NITS است نه الزام طراحان آن -م.

تابع *rijndael* سه پارامتر دارد که عبارتند از: (۱) *plaintext*: یک آرایه ۱۶ بایتی محتوی داده‌های ورودی؛ (۲) *ciphertext*: یک آرایهٔ ۱۶ بایتی که نهایتاً محتوی خروجی رمز شده را باز می‌گرداند. (۳) *key*: کلید رمز ۱۶ بایتی (۱۲۸ بیتی). در خلال پردازش، حالت فعلی داده‌ها در یک آرایهٔ دوبعدی به نام *state* که اندازهٔ آن *NROWS×NCOLS* است حفظ می‌شود. برای بلوکهای ۱۲۸ بیتی داده این آرایه ۴×۴ (یعنی ۱۶ بایتی) است. در این ۱۶ بایت، یک بلوک ۱۲۸ بیتی داده قابل ذخیره خواهد بود.

در ابتدا آرایهٔ *state* با داده‌های رمز نشدهٔ ورودی، مقداردهی اولیه می‌شود و سپس در هر مرحله از محاسبات مقدار آن تغییر خواهد کرد. در برخی از مراحل فقط جانشینی بایت به بایت انجام می‌شود و در برخی دیگر بر روی محتویات این آرایه جایگشت بایتها انجام می‌شود. البته (بغیر از جانشینی و جایگشت) تبدیلات دیگری نیز بر روی بایتها انجام می‌گیرد. در نهایت محتویات آرایهٔ *state* به عنوان متن رمز شده باز خواهد گشت.

این برنامه با توسعهٔ کلید ۱۲۸ بیتی به یازده آرایهٔ متفاوت و هم اندازه با آرایهٔ *state* کار خود را آغاز می‌کند؛ این یازده کلید در آرایهٔ *rk* ذخیره می‌شوند. *rk* آرایه‌ای از استراکچر است که به زبان C تعریف شده و هر عضو این آرایه خودش از نوع یک آرایهٔ *state* (۴×۴ بایتی) است. یکی از این کلیدها در بدو شروع محاسبات مورد استفاده قرار می‌گیرد و از ده تای دیگر در خلال ده دور پردازش بهره گرفته می‌شود. به هر یک از این ده کلید، «**کلید دور**» (Round Key) گفته می‌شود. محاسبه و استخراج کلیدهای دور بسیار پیچیده است و برای پرهیز از پیچیده شدن اصل موضوع در اینجا بدان نخواهیم پرداخت؛ چراکه از عمومیت کار کاسته نخواهد شد و تشریح آن برای اصل قضیه محوری نیست. فقط به ذکر این نکته بسنده می‌کنیم که کلیدهای هر «دور» براساس چرخش کلید (Rotation) و XOR کردن آن با گروههای خاصی از بیتهای خودِ کلید انجام می‌شود. برای تشریح کامل روش به مرجع (Daemen & Rijmen 2002) مراجعه نمائید.

گام بعدی آن است که متن اصلی در درون آرایهٔ *state* کپی شود تا در خلال ده دور متوالی پردازش شود. عمل کپی درون این آرایه به صورت ستونی انجام می‌شود یعنی اولین چهار بایت، در ستون اول (ستون شمارهٔ صفر)، چهار بایت دوم در ستون دوم (ستون شمارهٔ ۱) کپی می‌شود و به همین ترتیب ادامه می‌یابد. شماره‌گذاری ستونها و سطرها از شمارهٔ صفر شروع شده است در حالی که مراحل پردازش (دورها) از شمارهٔ یک شماره‌گذاری می‌شوند. مراحل تنظیم مقداردهی مقدماتی آرایه‌های ۴×۴ در شکل ۸-۱۰ نشان داده شده است.



شکل ۸-۱۰. ایجاد آرایه‌های state و rk.

قبل از شروع محاسبات اصلی، یک کار دیگر نیز انجام می‌شود: [0]*rk* با آرایهٔ *state* بایت به بایت *XOR* می‌شود. به عبارت دیگر هر یک از ۱۶ بایت آرایهٔ *state*، با مقدار حاصل از *XOR* خودش با بایت متناظر در [0]*rk* تعویض می‌گردد.

حال زمان شروع محاسبات اصلی فرا رسیده است. حلقه ده بار تکرار می‌شود (یکبار به ازای هر دور) و در هر تکرار محتوای *state* تغییر می‌کند. هر دور شامل چهار گام است: در گام اول محتوای *state* بایت به بایت با مقادیر جدید «جانشین» (Substitute) می‌شود. در حقیقت محتوای هر بایت به عنوان اندیس ورودی به یک S-box اعمال می‌شود تا خروجی معادل با خود را تولید نماید. این مرحله یک سیستم جانشینی ساده و کاراکتر به کاراکتر (monoalphabetic) است که در ابتدای این فصل بدان پرداختیم. برخلاف DES که چندین S-box مختلف دارد، در سیستم Rijndael فقط یک S-box وجود دارد.

گام دوم از هر دور، چهار سطرِ آرایهٔ *state* را به سمت چپ می‌چرخاند: سطر شماره صفر، صفر بایت می‌چرخد (یعنی تغییر نمی‌کند)، سطر شمارهٔ یک، یک بایت به سمت چپ می‌چرخد، سطر شمارهٔ دو، دو بایت و سطر شمارهٔ سه، سه بایت. در این گام محتوای بلوک فعلی آرایه بهم ریخته می‌شود که معادل با بلوک جایگشت در شکل ۸-۶-الف است.

در گام سوم هر ستون (از آرایهٔ *state*) بطور مستقل از دیگری درهم ریخته می‌شود. این عمل براساس ضرب ماتریسی انجام می‌گیرد، بدین نحو که ستون فعلی در یک ماتریس ثابت ضرب شده و ستون جدید را تولید می‌کند. عمل ضرب ماتریس مبتنی بر نظریه **«میدان محدود گالوا»**[1] یعنی $GF(2^8)$ انجام می‌شود. اگرچه این فرآیند ممکن است پیچیده به نظر برسد ولی یک الگوریتم ساده در این خصوص وجود دارد که در آن هر یک از ستونهای جدید براساس دو جستجو در جدول نگاشت[2] و سه عمل XOR محاسبه می‌شوند. برای کسب اطلاعات تفصیلی به مرجع (Rijndael, Daemen; 2002) مراجعه کنید.

نهایتاً در گام چهارم **«کلید دور»** با آرایهٔ *state* بایت به بایت XOR می‌شود.

از آنجایی که یکایک مراحل به سادگی برگشت‌پذیر هستند، لذا عمل رمزگشایی با اجرای برعکس الگوریتم [از آخر به اول] انجام می‌شود. با این وجود یک راه زیرکانه وجود دارد که در آن عمل رمزگشایی با اجرای همان الگوریتم رمزنگاری ولی با جداول متفاوت انجام می‌گیرد.

این الگوریتم هم امنیت بسیار بالا و هم سرعت بسیار عالی را تضمین می‌کند. پیاده‌سازی نرم‌افزاری آن بر روی یک ماشین 2-GHz می‌تواند عمل رمزنگاری هفتصد مگابیت داده در ثانیه را به صورت بلادرنگ انجام بدهد که برای رمزنگاری صد کانال ویدیویی MPEG-2 به صورت همزمان کفایت می‌کند. پیاده‌سازی سخت‌افزاری، از این هم سریعتر خواهد بود.

۸-۲-۳ حالات رمز (Cipher Modes)

علیرغم تمام پیچیدگیها، AES (یا DES یا هر سیستم رمزنگار که بر روی بلوک محدودی از داده‌ها عمل می‌کند) براساس سیستم رمز جانشینی بنیان گذاشته شده است که در آن یک بلوک بزرگ از کاراکترها (۱۲۸ بیتی در AES و ۶۴ بیتی در DES) با یک بلوک جدید جایگزین می‌شود. هرگاه بلوکهای مشابه از اطلاعات رمز نشده به ورودی این سیستم اعمال شود بلوکهای رمز شده یکسانی تولید خواهد شد. مثلاً اگر شما متن abcdefgh را با یک کلید مشابه در سیستم DES صد بار رمز کنید، صد نتیجهٔ یکسان خواهید گرفت. یک رمزشکن می‌تواند از این ویژگی برای واژگون کردن رمز (استخراج اطلاعات بدون در اختیار داشتن کلید) سوءاستفاده کند.

حالت کتابچهٔ رمز (Electronic Code Book Mode)

برای آن که ببینیم در رمزهای جانشینی چگونه از خصوصیت فوق‌الذکر برای شکستن رمز سوءاستفاده می‌شود،

---

۱. Finite Galois Field		۲. جستجو از طریق Lookup Table

سیستم رمز «DES سه‌گانه» (Triple DES) را بررسی می‌کنیم زیرا نشان دادن بلوکهای ۶۴ بیتی داده از بلوکهای ۱۲۸ بیتی ساده‌تر است ولیکن سیستم AES نیز دقیقاً همین مشکل را دارد. ساده‌ترین راه برای رمزنگاری یک قطعه طولانی داده آن است که به قطعات متوالی هشت بایتی (۶۴ بیتی) تقسیم شده و هر یک از این قطعات با کلید مشابه، یکی پس از دیگری رمزنگاری شوند. آخرین قطعه، ممکن است نیاز به اضافه کردن اطلاعات زائد (تا رسیدن به ۸ بایت) داشته باشد. این روش (یعنی قطعه قطعه کردن داده‌ها به بخشهائی با طول ثابت) ECB Mode[1] نامیده می‌شود که مشابه با یک سیستم قدیمی رمزنگاری است که در آن کلمات یک متن تفکیک شده و به جای آنها بکمک کتابچهٔ رمز یک کد پنج رقمی (در مبنای ده) جایگزین می‌شد تا متن رمز شده را ایجاد نماید.

در شکل ۸-۱۱، ابتدای یک فایل کامپیوتری را مشاهده می‌کنید که در آن فهرستی از پاداشهای سالانهٔ یک شرکت که قرار است به کارمندان خود اعطاء کند، درج شده است. این فایل از رکوردهای متوالی ۳۲ بایتی تشکیل شده و به ازای هر کارمند یک رکورد، طبق قالب ذیل ذخیره شده است: ۱۶ بایت برای نام، ۸ بایت برای ردهٔ شغلی و ۸ بایت برای پاداش. فایل مربوطه که جمعاً شامل ۱۶ بلوک ۸ بایتی است که از شمارهٔ صفر تا ۱۵ شماره‌گذاری شده و بکمک «سیستم DES سه‌گانه» (Triple DES) رمزنگاری شده است.

نام		جایگاه شغلی	پاداش
Adams,	Leslie	Clerk	$10
Black,	Robin	Boss	$500,000
Collins,	Kim	Manager	$100,000
Davis,	Bobbie	Janitor	$5
Bytes ← 16 →		← 8 →	← 8 →

شکل ۸-۱۱. متن اصلی از یک فایل که در قالب ۱۶ بلوک بروش DES رمز شده است.

فرض کنید Leslie با رئیسش مشکل دارد و طبعاً انتظار دریافت پاداش چندانی در سر نمی‌پرورد. در طرف مقابل Kim مورد توجه و عنایت رئیس است و همه این را می‌دانند. Leslie می‌تواند به این فایل بعد از رمز شدن ولی قبل از ارسال به بانک دسترسی پیدا کند. آیا Leslie قادر است وضعیت نامناسب پاداش خود را (در حالی که اطلاعات درون فایل رمز شده است) اصلاح نماید؟

هیچ مشکلی برای این کار وجود ندارد! تمام کاری که Leslie باید انجام بدهد آن است که کپی بلوک دوازدهم از متن رمز شده (که شامل پاداش Kim است) را استخراج کرده و آنرا با بلوک چهارم (رکورد پاداش خودش) عوض کند. حتی بدون آن که Leslie بداند در بلوک دوازدهم چه چیزی درج شده است، قطعاً انتظار ایام بسیار شادتری را در کریسمس این سال خواهد داشت! (البته او می‌تواند بلوک رمز شدهٔ هشتم ـ پاداش رئیس ـ را برای خودش کپی کند ولیکن به احتمال زیاد قضیه فاش خواهد شد!! گذشته از آن Leslie آدم حریص و زیاده‌طلبی نیست!)

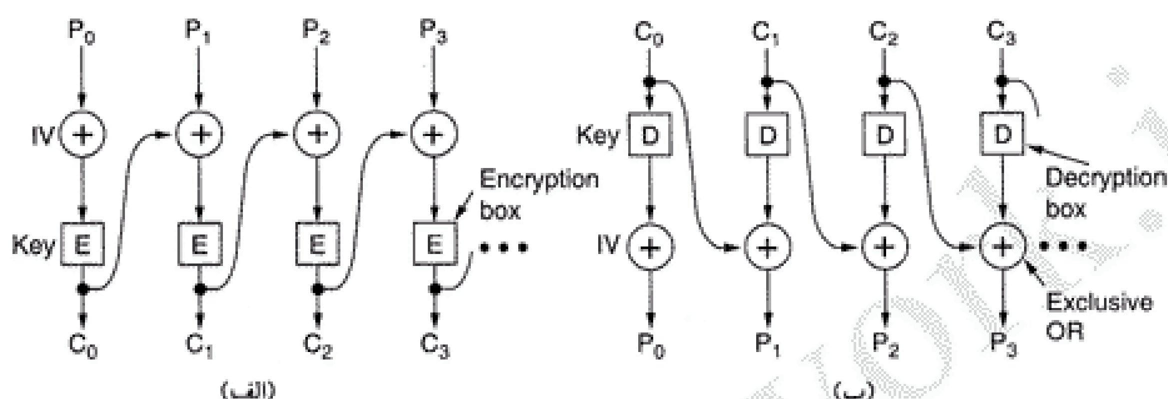
حالت زنجیره‌سازی بلوکهای رمز (Cipher Block Chaining Mode)

برای خنثی کردن این نوع حملات، بلوکهای داده می‌توانند به صورت زنجیره‌ای رمز شوند به گونه‌ای که تغییر یا جابجایی در یک بلوک (همانند کاری که Leslie انجام داد) باعث شود متن رمزگشایی شده از محل دستکاری، به بلوکهای آشغال و بی‌معنی تبدیل گردد. یکی از روشهای زنجیره‌سازی، اصطلاحاً روش

---

۱. Electronic Code Book Mode

Cipher Block Chaining نام دارد. در این روش که در شکل ۸-۱۲ نشان داده شده هر بلوک از اطلاعات اصلی، پیش از رمزنگاری با بلوک رمز شدهٔ قبل از خودش XOR می‌شود. بدین ترتیب بلوکهای یکسان متن به بلوکهای رمز شدهٔ مشابه تبدیل نخواهد شد و رمزنگاری از حالت «جانشینی بلوک» خارج خواهد گردید. اولین بلوک با یک مقدار تصادفی به نام IV (Initialization Vector)، XOR می‌شود و به صورت آشکار به همراه داده‌های رمز شده ارسال می‌گردد.



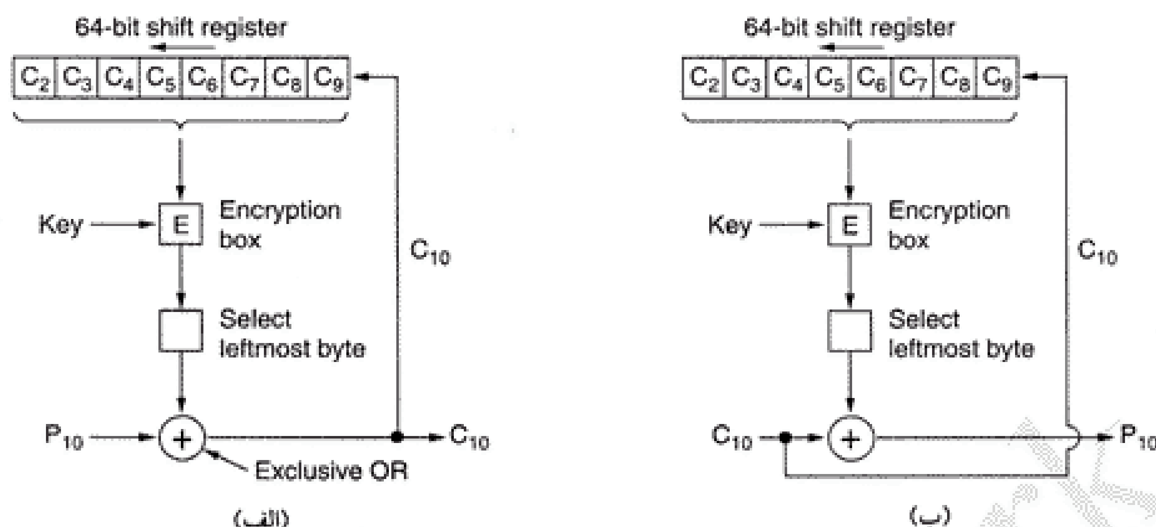
شکل ۸-۱۲. زنجیره سازی بلوکهای رمز. (الف) رمزنگاری. (ب) رمزگشایی.

با بررسی شکل ۸-۱۲ به سادگی می‌توان چگونگی زنجیره‌سازی بلوکهای رمز را مشاهده و بررسی کرد. کار را با محاسبهٔ $C_0=E(P_0 \text{ XOR } IV)$ شروع می‌کنیم. سپس بلوک بعدی رمز را با محاسبهٔ $C_1=E(P_1 \text{ XOR } C_0)$ بدست آورده و به همین ترتیب ادامه می‌دهیم. دقت داشته باشید که عمل رمزگشایی بلوک i، منوط به آن است که تمام بلوکهای صفر تا i-1 از رمز خارج شده باشند بدین ترتیب بلوکهای مشابه در متن اصلی بسته به محل قرار گرفتنشان، بلوکهای رمز شدهٔ کاملاً متفاوتی را ایجاد خواهند کرد. اگر تبدیل یا تغییری در فایل رمز شده انجام گیرد (از نوعی که Leslie در مثال بالا انجام داد)، پس از رمزگشایی فایل، از محل دستکاری به بعد بلوکها بی‌معنی و نامعتبر خواهند بود. برای یک کارآگاه امنیتی زیرک، محلی که از آنجا به بعد این ناهنجاری بوجود آمده (یعنی محلی که پس از رمزگشایی، بلوکها بی‌معنی شده‌اند) می‌تواند نقطهٔ شروع خوبی برای آغاز بازرسی و پیگیری باشد!

زنجیره سازی بلوکهای رمز [گذشته از خنثی کردن حملاتی نظیر آنچه که در بالا اشاره شد] این حسن بزرگ را دارد که بلوکهای مشابه متن، بلوکهای رمز یکسان تولید نخواهند کرد و بالطبع کار تحلیل‌گر رمز [برای شکستن رمز] بسیار سخت‌تر خواهد شد. در حقیقت این حسن دلیل اصلی استفاده از آن می‌باشد.

حالت فیدبک رمز (Cipher Feedback Mode)

با وجود این، زنجیره سازی بلوکهای رمز یک اشکال دارد و آن هم این که تا موقعی که یک بلوک ۶۴ بیتی بطور کامل دریافت نشود، رمزگشایی آن [و بلوکهای بعدی حتی در صورت دریافت] ممکن نخواهد بود. استفاده از این روش در یک ترمینال محاوره‌ای که در آن جا کاربران می‌توانند خطوط یا فرامین کوتاهتر از هشت کاراکتر درج و ارسال کنند و برای دریافت پاسخ منتظر بمانند، چندان مناسب نیست. برای رمزنگاری بایت‌به‌بایت (به نحوی که در شکل ۸-۱۳ می‌بینید) از روش Cipher Feedback Mode مبتنی بر رمزنگاری «DES سه‌گانه» (Triple DES) استفاده می‌شود. برای روش AES نیز دقیقاً همین ایده کارساز خواهد بود با این تفاوت که در آنجا شیفت رجیسترها ۱۲۸ بیتی هستند. در این شکل وضعیت ماشین رمزنگار در حالتی نشان داده شده که قبل از آن بایتهای صفر تا ۹، رمز و ارسال شده‌اند. وقتی بایت دهم از راه می‌رسد، (مطابق با شکل ۸-۱۳-الف) الگوریتم DES بر روی محتوای ۶۴ بیتی موجود در شیفت رجیستر اعمال شده و کدهای رمز ۶۴ بیتی در خروجی آماده



شکل ۸–۱۳. حالت فیدبک رمز. (الف) رمزنگاری. (ب) رمزگشایی.

می‌شوند. سپس بایت سمت چپ این متن رمزی استخراج و با بایت دهم تازه وارد [یعنی P_{10}]، XOR می‌شود و بایت حاصل یعنی C_{10} بر روی خط انتقال ارسال خواهد شد. بعلاوه وقتی C_{10} ارسال شد، شیفت رجیستر، بایتها را به سمت چپ شیفت داده و C_{10} در سمت راست شیفت رجیستر قرار می‌گیرد. دقت کنید که محتوای فعلی شیفت رجیستر به پیشینهٔ کل متن ارسالی وابسته خواهد بود، بدین ترتیب یک الگوی تکراری در متن اصلی به صور گوناگون در متن رمز شده نگاشته می‌شود. دقیقاً همانند روش زنجیره سازی بلوکهای متن، در این روش نیز برای شروع گردش عملیات به یک مقدار اولیه (Initialization Vector) نیاز خواهد بود.

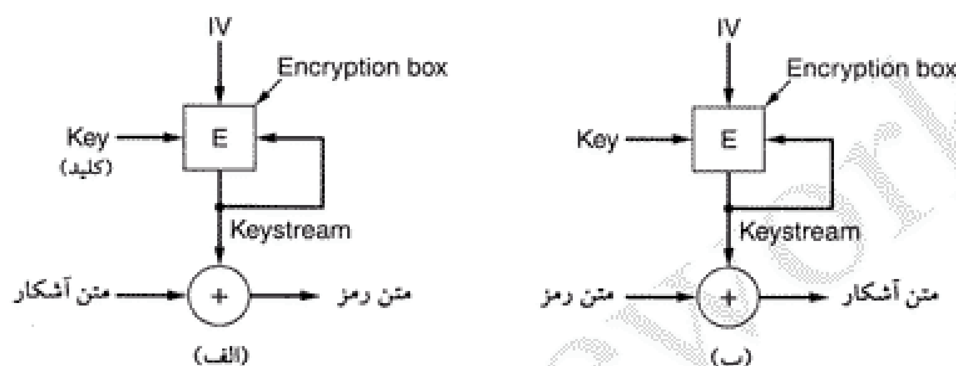
رمزگشایی در روش Cipher Feedback Mode دقیقاً مشابه با عملیات رمزنگاری داده‌ها است. در اصل محتوای شیفت رجیستر به جای رمزگشایی مجدداً رمزنگاری می‌شود زیرا هر گاه خروجی رمز شده شیفت رجیستر، با C_{10} XOR شود P_{10} را نتیجه خواهد داد چراکه در ابتدا برای بدست آمدن C_{10} خروجی رمز شده شیفت رجیستر با P_{10}، XOR شده بود [و XOR مجدد آن با C_{10} کاراکتر P_{10} را نتیجه خواهد داد.] تا زمانی که دو رجیستر مشابه هم هستند عمل رمزگشایی به درستی انجام می‌شود. این مفهوم در شکل ۸–۱۳–ب نشان داده شده است.

مشکلی که در روش Cipher Feedback Mode وجود دارد آن است که هر گاه یک بیت از متن رمز شده در خلال ارسال وارونه شود، تا زمانی که این بیت خراب در درون شیفت رجیستر قرار دارد نتیجهٔ رمزگشایی غلط خواهد بود و بدین ترتیب به ازای هر بیت خطا، هشت بایت اشتباه در خروجی پدیدار خواهد شد. هر گاه بیت اشتباه از شیفت رجیستر بیرون بیفتد مجدداً خروجی صحیح تولید خواهد شد. بنابراین تأثیر یک بیت اشتباه کاملاً محلی است و به مابقی پیام سرایت نخواهد کرد بلکه فقط بیتهایی به پهنای شیفت رجیستر را خراب می‌کند.

Stream Cipher Mode

علیرغم محلی بودن اثر یک بیت خطا در روش قبل، برنامه‌های کاربردی خاصی وجود دارند که در آنها سرایت خطای یک بیت به ۶۴ بیت، تأثیر مخرب بسیار زیاد، بجا می‌گذارد. برای این نوع از برنامه‌های کاربردی گزینهٔ چهارمی به نام Stream Cipher Mode وجود دارد. این روش با رمز کردن یک مقدار اولیه به نام بردار IV (Initialization Vector) توسط یک کلید کارش را شروع می‌کند تا یک بلوک خروجی بدست آید. بلوک

خروجی مجدداً رمز می‌شود تا بلوک دوم بدست آید. بلوک دوم نیز مجدداً رمز شده تا بلوک سوم بدست آید و این روال ادامه خواهد یافت. دنبالهٔ (طولانی و اختیاری) بلوکهای خروجی که اصطلاحاً Keystream نامیده می‌شود با بلوکهای متن، XOR می‌گردند و بدین ترتیب همانند آنچه که در شکل ۸-۱۴ ملاحظه می‌کنید متن رمز می‌شود.[۱] دقت کنید که IV فقط در مرحلهٔ اول مورد استفاده قرار می‌گیرد؛ پس از آن، خروجی هر مرحله رمز می‌شود تا Keystream جدید بدست آید. همچنین توجه کنید که Keystream مستقل از داده‌هاست لذا حتی می‌توان Keystream هر مرحله را با داشتن کلید و بردار IV پیشاپیش محاسبه کرد و بنابراین Keystream نسبت به خطاهایی که در حین انتقال ممکن است اتفاق بیفتد حساس نیست.[۲] عمل رمزگشایی در شکل ۸-۱۴-ب نشان داده شده است.



شکل ۸-۱۴. حالت استریم رمز (Stream Cipher Mode). (الف) رمزنگاری. (ب) رمزگشایی.

رمزگشایی در سمت گیرنده با تولید Keystreamهای مشابه انجام می‌شود. از آنجایی که Keystream فقط به IV و کلید رمز وابسته است، لذا از خطاهای احتمالی که در حین انتقال برای برخی از بیتهای متن رمز شده اتفاق می‌افتد، تأثیر نمی‌گیرد. بدین ترتیب یک بیت خطا در حین انتقال متن رمز شده، فقط و فقط یک بیت خطا در متن رمزگشایی شده ایجاد خواهد کرد.

نکتهٔ حیاتی آنست که در این روش هیچگاه نباید از زوج (**کلید** ، **بردار** IV) مشابه استفاده شود زیرا این کار منجر به تولید Keystreamهای یکسان خواهد شد. استفاده از Keystreamهای مشابه، متن رمز شده را در معرض آسیب حملهٔ *Keystream reuse attack* قرار می‌دهد: فرض کنید یک بلوک از متن اصلی، مثلاً P_0 ، طبق قاعدهٔ P_0 XOR K_0 رمز شده باشد. [که در آن P_0 بلوک اول از متن اصلی و K_0 همان Keystream تولید شده در مرحلهٔ اول است.] همچنین فرض کنید بعداً برای متن جدید Q نیز از همین Keystream استفاده شود. بدین ترتیب بلوک Q_0 از متن دوم نیز طبق قاعدهٔ Q_0 XOR K_0 رمز می‌شود. یک رمزشکن که توانسته بلوکهای رمز شدهٔ پیامهای P و Q را جداگانه در اختیار بگیرد، این دو بلوک را با هم XOR می‌کند و با اینکار کلید از میان می‌رود.[۳] اگر یکی از بلوکهای P_0 یا Q_0 معلوم باشد یا حدس زده شود، دیگری هم به سادگی بدست می‌آید. به هر تقدیر می‌توان به حاصل XOR دو متن اصلی، طبق روشها و ویژگیهای آماری حمله کرد و آنها را آشکار ساخت. به عنوان مثال در متن انگلیسی، دو کاراکتر فاصله خالی (Space) بیشترین احتمال XOR شدن را دارند. پس از آن XOR شدن

---

۱. در حقیقت این روش همانند روش رمزنگاری One Time Pad که در ابتدای این فصل تشریح شد عمل می‌کند با این تفاوت که Pad به صورت خودکار، توسط کلید و IV تولید می‌شود. -م

۲. در حقیقت رمزنگاری پیام فقط یک عمل XOR ساده با Keystreamها در هر مرحله است. -م

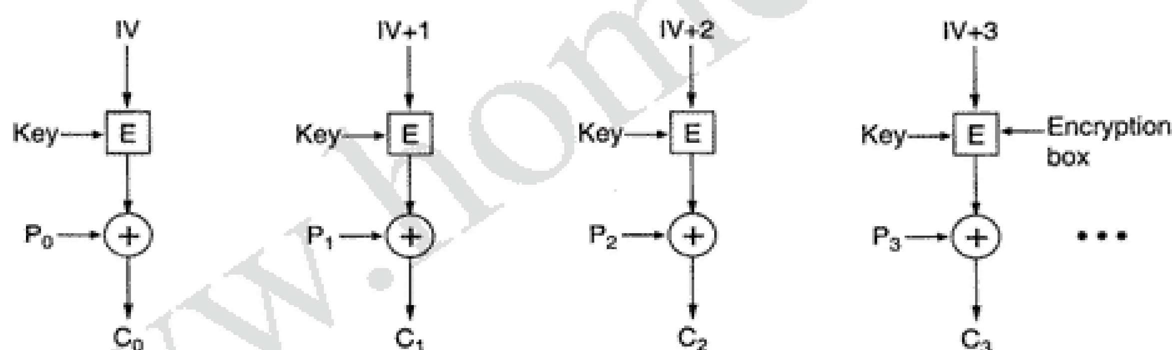
۳. عمل XOR این دو بلوک رمز عبارت است از (P_0 XOR K_0) XOR (Q_0 XOR K_0) که معادل با P_0 XOR Q_0 خواهد شد. -م

حرف 'e' با Space بالاترین احتمال را دارد و به همین ترتیب. کوتاه سخن آنکه، با در اختیار داشتن حاصل XOR شدهٔ دو متن ، رمزشکن اقبال بسیار بلندی برای آشکار کردن هر دوی آنها خواهد داشت.

حالت شمارنده (Counter Mode)

یکی از مشکلاتی که تمام روشهای قبل، به استثنای روش Electronic Code Book، دارند آنست که دسترسی تصادفی به داده‌های رمز شده ممکن نیست. به عنوان مثال فرض کنید یک فایل بر روی شبکه ارسال و به صورت رمز شده بر روی دیسک ذخیره گردد. ذخیرهٔ فایلها به صورت رمز شده از این دیدگاه که ماشین گیرندهٔ فایل یک کامپیوتر کیفی است و احتمال دارد سرقت شود، کاملاً منطقی است . ذخیرهٔ فایلها به صورت رمز شده خطر فاش شدن اطلاعات را کاهش خواهد داد مخصوصاً زمانی که احتمال دارد یک کامپیوتر بدست افراد ناباب بیفتد.

ولیکن در عمل، دسترسی به فایلهای روی دیسک و بالاخص پایگاههای اطلاعاتی، به صورت غیرترتیبی (nonsequential) انجام می‌شود. برای دسترسی به یک بلوک تصادفی از فایلی که به روش «زنجیره‌سازی بلوکها»[1] رمزنگاری شده است باید ابتدا تمام بلوکهای قبل از آن رمزگشایی شوند که این فرآیند (از لحاظ زمان دسترسی) بسیار پر هزینه است.[2] به همین دلیل روش دیگری ابداع شده که ساختار آنرا در شکل ۸–۱۵ مشاهده می‌کنید. در این روش متن اصلی به صورت مستقیم رمز نمی‌شود بلکه یک بردار اولیه (IV) که با یک عدد صحیح جمع می‌شود توسط کلید رمز شده و نتیجهٔ آن با متن اصلی XOR می‌شود. برای هر بلوک جدید یک واحد به IV اضافه می‌شود فلذا دسترسی به هر بلوک از داده در هر کجای آن بدون نیاز به رمزگشایی بلوکهای قبلی آن، میسر خواهد بود.



شکل ۸–۱۵. رمزنگاری در حالت شمارنده (Counter Mode)

اگرچه روش Counter Mode سودمند است ولیکن از یک ضعف اساسی رنج می‌برد که اشاره به آن خالی از لطف نیست. فرض کنید که در آینده [برای رمزنگاری فایلی دیگر] از کلیدی مشابه K استفاده شود (با IV مشابه ولی با متنی متفاوت) و یک رمزشکن این دو متن رمز شده را بدست بیاورد. چون کلید و IV مشابهند بنابراین Keystreamها [یعنی کلیدهایی که برای هر بلوک جدید محاسبه و با آن XOR می‌شوند] یکسانند و طبعاً سیستم رمز Counter Mode در معرض حملهٔ *Keystream Reuse Attack* که در بخش قبلی تشریح شد، قرار می‌گیرد. رمزشکن با XOR کردن یکایک بلوکهای این دو متن، کلید رمز را از میان برمی دارد و حاصل XOR شدهٔ دو متن را بدست می‌آورد. این ضعف بدان معنا نیست که روش Counter Mode، نظریهٔ بدی است، بلکه بدین مفهوم

---

۱. Cipher Block Chaining

۲. به عبارت بهتر دسترسی مستقیم به یک بلوک از چنین فایلی هرگز میسر نیست مگر آن که تمام بلوکهای قبل از آن، از رمز خارج شده باشند. ــم

است که چه کلید و چه بردار IV باید مستقل از هم و به صورت کاملاً تصادفی انتخاب شود. در این صورت وقتی IV انتخابی متفاوت باشد، حتی اگر از یک کلید مشابه به صورت اتفاقی دو بار استفاده شود، متن رمز شده امن خواهد ماند.

۸-۲-۴ رمزهای دیگر

DES و Rijndael، مشهورترین الگوریتمهای رمزنگاری با کلید متقارن هستند ولیکن باید به این نکته دقت داشت که روشهای متعدد دیگری برای رمزنگاری با کلید متقارن ابداع شده است. برخی از این الگوریتمها در درون تولیدات مختلف [مثل کارتهای هوشمند، کدکننده‌های ویدیویی یا برخی از نرم‌افزارها] پیاده‌سازی شده‌اند. فهرست رایجترین این روشها را در شکل ۸-۱۶ می‌بینید.

نام رمز	ابداع کننده	طول کلید	توضیح
Blowfish	Bruce Schneier	1–448 bits	قدیمی است و کند عمل می‌کند.
DES	IBM	56 bits	برای کاربردهای امروزی بسیار ضعیف است.
IDEA	Massey and Xuejia	128 bits	روش مناسبی است ولی امتیاز آن ثبت شده
RC4	Ronald Rivest	1–2048 bits	احتیاط : برخی از کلیدها ضعیف عمل می‌کنند!
RC5	Ronald Rivest	128–256 bits	روش مناسبی است ولی امتیاز آن ثبت شده
Rijndael	Daemen and Rijmen	128–256 bits	بهترین گزینه ممکن
Serpent	Anderson, Biham, Knudsen	128–256 bits	بسیار محکم و قوی
Triple DES	IBM	168 bits	دومین گزینه مناسب
Twofish	Bruce Schneier	128–256 bits	بسیار قوی است و کاربرد گسترده‌ای دارد.

شکل ۸-۱۶. برخی از الگوریتمهای رایج رمزنگاری با کلید متقارن

۸-۲-۵ تحلیل رمز (رمزشکنی)

قبل از آن که موضوع رمزنگاری با کلید متقارن را خاتمه بدهیم، اشاره به چهار روش توسعه یافته در تحلیل رمز (رمزشکنی) ارزشمند خواهد بود. اولین روش، «**تحلیل رمز به روش تفاضلی**» (Differential Cryptoanalysis) است. (توسط Biham و Shamir، ۱۹۹۳) این روش می‌تواند برای حمله به هر سیستم رمز بلوکی به کار گرفته شود. در اینجا، کار تحلیل رمز با انتخاب دو بلوک متن که در تعداد بسیار کمی بیت با هم اختلاف دارند، آغاز می‌شود؛ سپس با اعمال آنها در الگوریتم رمزنگاری، اتفاقات حاصل از اجرای هر مرحله از الگوریتم، بدقت بررسی می‌شود. در بسیاری از حالات، برخی از الگوهای بیتی نسبت به الگوهای دیگر بیشتر تکرار می‌شوند و این نتیجه و مشاهده می‌تواند در هدایت حملهٔ مبتنی بر احتمال و آمار بسیار مؤثر باشد.

دومین روش، «**تحلیل رمز خطی**»[1] است (Matsui, 1994) که می‌تواند رمز DES را با آزمایش 2^{43} حالت شناخته شدهٔ متن، بشکند. این روش با XOR کردن بیتهایی مشخص از متن اصلی و متن رمز شده با یکدیگر و آزمایش مجدد الگوی بدست آمده انجام می‌شود. هر گاه این روال به تعداد دفعات زیاد تکرار شود، نیمی از بیتها باید صفر و نیم دیگر یک باشند.[2] ولیکن اغلب نتیجه بدست آمده، نسبت به یکی از جهتها (یعنی بیتهای صفر و

---

۱. Linear Cryptanalysis

۲. طبق نظریهٔ احتمال، هر گاه متن ورودی هیچگاه به کلید وابستگی آماری نداشته باشد، احتمال ظاهر شدن صفرها و یکها مساوی ۰/۵ است. ـم

یک) «بایاس» پیدا می‌کند؛[1] این بایاس حتی اگر کوچک باشد می‌تواند برای کاهش Work Factor (جستجوی فضای کلید) مورد سوءاستفاده واقع شود. برای شرح کامل این روش به مقالهٔ Matsui مراجعه نمائید.

سومین روش برای تحلیل رمز، استفاده از میزان توان الکتریکی مصرفی پردازنده برای یافتن کلید سرّی است. بطور معمول کامپیوترها از ولتاژ ۳ ولت برای نمایش بیت یک و از ولتاژ صفر برای بیت صفر استفاده می‌کنند. بنابراین پردازش بیت ۱ انرژی الکتریکی بیشتری نسبت به بیت صفر مصرف می‌کند. اگر یک الگوریتم رمزنگاری شامل حلقه‌ای باشد که در آن بیتهای کلید به ترتیب پردازش می‌شوند، رمزشکن سیگنال ساعت n گیگاهرتزی را به مقدار کمتری کاهش داده (مثلاً صد مگاهرتز) و یک گیرهٔ کوچک از نوع سوسماری را به پایه‌های تغذیه و زمین CPU متصل کرده و بر توان مصرفی پردازنده در حین اجرای هر دستورالعمل نظارت می‌کند. استنتاج کلید رمز از این اطلاعات بسیار ساده خواهد بود. برای خنثی کردن این نوع از رمزشکنی، می‌توان الگوریتم رمزنگاری را به زبان اسمبلی برنامه‌نویسی کرده و مطمئن شد که توان مصرفی پردازنده مستقل از بیتهای کلید و همچنین «کلیدهای دور» (Round Keys) است.

چهارمین روش، «تحلیل زمانی» (Time Analysis) است. الگوریتمهای رمزنگاری سرشار از دستورات if then else هستند که در حقیقت بیتهایی از کلیدهای هر دور را آزمایش می‌کنند. هر گاه دستورات پس از then و else از لحاظ زمان اجرا، متفاوت باشند، با پایین آوردن سرعت سیگنال ساعت پردازنده و اندازه‌گیری زمان اجرای هر مرحله، استنتاج کلیدهای هر دور ممکن خواهد بود. وقتی یکایک کلیدهای هر دور از اجرای الگوریتم بدست آمد می‌توان کلید اصلی را محاسبه کرد. تحلیل توان مصرفی و تحلیلهای زمانی را می‌توان بطور همزمان بکار گرفت تا کار کشف کلید رمز ساده‌تر شود. اگرچه تحلیل توان و زمان ممکن است عجیب به نظر برسد ولی در واقع این دو روش قادرند هر گونه سیستم رمز را که در مقابل چنین حمله‌ای، مقاوم طراحی نشده باشد، بشکنند.

۸ـ۳ الگوریتمهای کلید عمومی (Public Key)

همواره توزیع و مبادلهٔ کلید رمز (Key Distribution) یکی از مشکلات سیستمهای رمزنگاری بوده است. فارغ از آن که یک سیستم رمزنگاری چقدر قدرتمند و محکم است، هرگاه یک اخلالگر بتواند کلید رمز را سرقت کند، کل سیستم بی‌ارزش خواهد شد. رمزشکنها همیشه از روشهایی که در آنها کلید رمزنگاری و رمزگشایی یکسان است (یا از طریق یکدیگر قابل محاسبه هستند) قلباً استقبال می‌کنند. در این روشها بالاخره باید کلیدها بین کاربران سیستم توزیع شود. در همین نقطه به نظر می‌رسد که یک اشکال ذاتی و درونی وجود دارد. از یک طرف این کلیدها باید در مقابل سرقت حفاظت شوند و از طرف دیگر باید بین کاربران توزیع شوند، بنابراین نمی‌توان از این کلیدها در گاوصندوق نگهداری کرد!

در سال ۱۹۷۶، دو پژوهشگر در دانشگاه استنفورد به نامهای دیفی و هلمن (۱۹۷۶) یک سیستم رمز کاملاً جدید را پیشنهاد کردند که در آن کلیدهای رمزنگاری و رمزگشایی متفاوت بودند و با در اختیار داشتن کلید رمزنگاری عملاً نمی‌شد کلید رمزگشایی را استنتاج کرد. در طرح پیشنهادی این دو نفر، الگوریتم رمزنگاری E (با کلید e) و الگوریتم رمزگشایی D (با کلید d)، باید سه نیاز را برآورده می‌کرد. این نیازها را می‌توان به سادگی به صورت زیر توصیف کرد:

۱. $D(E(P))=P$
۲. استنتاج d (کلید رمزگشایی) از روی e (کلید رمزنگاری) بی‌نهایت مشکل باشد.
۳. E از طریق مکانیزم «حمله با متن‌های انتخابی و شناخته شده» شکسته نشود.

---

۱. یعنی احتمال وقوع یکی از بیتها از ۰/۵ کمتر و دیگری از ۰/۵ بیشتر می‌شود. ـم

اولین نیاز بیانگر آن است که هر گاه الگوریتم رمزگشایی D را بر روی یک متن رمز شده یعنی $E(P)$ اعمال کنیم مجدداً اصل پیام P را بدست بیاوریم. بدون این ویژگی گیرندهٔ مجاز نیز قادر به رمزگشایی متن رمزی نخواهد بود. نیاز دوم به قدر کافی گویاست و احتیاجی به توضیح اضافی ندارد. نیاز سوم به نحوی که بعداً خواهیم دید از آن جهت است که یک رمزشکن ممکن است الگوریتم را با استفاده از متنهای شناخته شده بیازماید و به روش سعی و خطا متن رمز شده را بشکند. با این سه شرط دلیلی وجود ندارد که کلید رمزنگاری را نتوان به صورت عمومی در اختیار همه قرار داد.

روش کار بدین نحوست که یک شخص مثلاً آلیس، وقتی تمایل دارد پیامهای محرمانه دریافت کند باید ابتدا دو الگوریتم منطبق با شرایط فوق ابداع کند. الگوریتم و کلید رمزنگاری آلیس به صورت عمومی و آشکار اعلام می‌شود. آلیس حتی می‌تواند کلید عمومی [برای رمزنگاری] را در صفحهٔ اصلی از وب‌سایت خودش به همه اعلام کند. ما از نماد E_A به معنای الگوریتم رمزنگاری با پارامتر A یعنی کلید عمومی آلیس، استفاده می‌کنیم. همچنین از نماد D_A به معنای الگوریتم رمزگشایی با پارامتر A یعنی کلید خصوصی آلیس، استفاده می‌نمائیم. باب نیز دقیقاً همین کار را می‌کند، E_B را به صورت عمومی آشکار می‌کند در حالی که D_B را به صورت سرّی نزد خود نگهداری می‌کند.

حال ببینیم مشکل برقراری یک کانال مطمئن بین آلیس و باب که هیچ ارتباط قبلی با هم نداشته‌اند چگونه حل می‌شود. فرض شده کلید رمزنگاری آلیس یعنی E_A و کلید رمزنگاری باب یعنی E_B در فایلهای قابل خواندن [و به صورت آشکار] قرار دارد. آلیس اولین پیام خود یعنی P را می‌گیرد و $E_B(P)$ را محاسبه کرده و نتیجه را برای باب می‌فرستد. باب با اعمال کلید سرّی خود یعنی D_B، آنرا رمزگشایی می‌کند. (به عبارت دیگر $D_B(E_B(P))$ را محاسبه می‌کند.) هیچ شخص دیگری نمی‌تواند از پیام رمزنگاری شده بهره‌برداری کند چراکه سیستم رمزنگاری بسیار قدرتمند فرض شده و استنتاج D_B (کلید رمزگشایی) از کلید رمزنگاری E_B بسیار مشکل و غیرعملی است. برای ارسال پاسخ پیام یعنی R، باب $E_A(R)$ را ارسال می‌کند. حال آلیس و باب می‌توانند به صورت مطمئن با یکدیگر مبادله پیام نمایند، بدون آن که کلیدهای سری آنها را غیر از خودشان کسی بداند.

شاید اشاره به چند اصطلاح در خصوص این روش مفید باشد. رمزنگاری با کلید عمومی (Public Key Cryptography) ایجاب می‌کند که هر کاربر دو کلید داشته باشد: یک **کلید عمومی** (Public Key) که تمام دنیا برای ارسال پیام به کاربر، از آن استفاده می‌کنند و یک **کلید خصوصی** (Private Key) که کاربر برای رمزگشایی پیامها بدان احتیاج دارد. از این به بعد، به کرّات از اصطلاحات کلید عمومی و خصوصی استفاده خواهیم کرد تا از **«کلید سرّی»** (Secret Key) که در سیستمهای رمزنگاری با کلید متقارن کاربرد دارد تمیز داده شود.

۸-۳-۱ RSA

تنها کاری که باید انجام شود یافتن الگوریتمی است که سه نیاز اشاره شده را برآورده نماید. به دلیل محاسن بالقوه‌ای که رمزنگاری با کلید عمومی در بردارد، بسیاری از پژوهشگران در این زمینه بشدت فعالیت می‌کنند و تاکنون چندین الگوریتم مناسب تدوین و معرفی شده است. یکی از الگوریتمهای خوب، توسط گروهی در دانشگاه MIT (به سرپرستی ری‌وِست، ۱۹۷۸) ابداع شد. این روش که به نام RSA مشهور است از حرف ابتدایی اسامی مخترعین آن، ری‌وِست، شامیر و آدِلْمن (Rivest, Shmir, Adelman) گرفته شده است. روش RSA برای حدود ربع قرن در مقابل تلاشهای فراوان برای شکستن آن، دوام آورده و یک روش بسیار قدرتمند تلقی می‌شود. بسیاری از روشهای عملی امنیت، براساس RSA هستند. بزرگترین اشکال این روش آن است که برای رسیدن به بالاترین درجهٔ امنیت، به کلید رمزی با حداقل ۱۰۲۴ بیت احتیاج است (برخلاف الگوریتمهای با کلید

متقارن ۱۲۸ بیتی) که این موضوع الگوریتم را بسیار کُند می‌کند.

روش RSA بر یک سری از اصول اساسی در نظریهٔ اعداد استوار است. ما چکیدهٔ این روش را در همین جا بیان می‌کنیم؛ برای تفصیل بیشتر از مقالات کمک بگیرید.

۱. دو عدد اول بسیار بزرگ p و q را انتخاب نمائید. (عموماً ۱۰۲۴ بیتی)
۲. حاصل n=p×q و Z = (p-۱).(q-۱) را بدست بیاورید.
۳. عددی انتخاب کنید که نسبت به Z اول باشد و آنرا d بنامید.
۴. e را به گونه‌ای پیدا کنید که e×d mod Z = ۱ برقرار باشد.

با این پارامترها که پیشاپیش محاسبه می‌شود، آمادهٔ شروع رمزنگاری هستیم: متن اصلی (که به صورت رشته‌ای از بیتها تلقی می‌شود) ابتدا به تعدادی بلوک تقسیم می‌گردد، به نحوی که هر بلوک P در بازهٔ $0 <= P < n$ قرار بگیرد. این کار را با تقسیم متن به بلوکهای k بیتی که در آن k بزرگترین عدد صحیحی است که در رابطهٔ $2^k < n$ صدق می‌کند، انجام بدهید. ($n=p\times q$)

برای رمزنگاری پیام P، $C = P^e \bmod n$ را محاسبه نمائید. برای رمزگشایی نیز $P = C^d \bmod n$ را حساب کنید. براحتی قابل اثبات است که توابع رمزگشایی و رمزنگاری، توابع معکوس یکدیگر هستند. برای رمزنگاری فقط به e و n احتیاج دارید. برای رمزگشایی به d و n نیازمندید. بنابراین کلید عمومی متشکل از (n و e) است و کلید خصوصی (n و d) خواهد بود.

امنیت این روش از آنجا ناشی شده که تجزیهٔ اعداد بسیار بزرگ به عوامل اول بسیار دشوار است. اگر رمزشکن بتواند عدد n را به عوامل اول تجزیه کند، قادر خواهد بود p و q را پیدا کرده و از این راه Z را محاسبه نماید. با در اختیار داشتن Z و e می‌توان توسط الگوریتم اقلیدس d را پیدا کرد. خوشبختانه، ریاضی‌دانها از حدود سیصد سال قبل برای تجزیهٔ اعداد بزرگ [به عوامل اول] تلاش کرده‌اند و شواهد حاکی از آن است که این کار بسیار مشکل می‌باشد.

براساس گزارش **ری‌وست** و گروه ابداع کنندهٔ RSA، تجزیهٔ یک عدد ۵۰۰ رقمی به روش «تکرار و آزمون» حدود 10^{25} سال طول می‌کشد. در هر حال آنها فرض را بر آن گذاشتند که بهترین الگوریتم و سریعترین کامپیوتر هر دستورالعمل را در یک میکروثانیه انجام بدهد. حتی اگر کامپیوترها در هر دهه به صورت نمایی سریعتر شوند باز هم تا زمانی که تجزیهٔ یک عدد ۵۰۰ رقمی امکان‌پذیر شود، قرنها باقی مانده است و در آن زمان هم می‌توان برای امن کردن RSA، p و q را بزرگتر انتخاب کرد!!

یک مثال بسیار سادهٔ آموزشی از عملکرد الگوریتم RSA در شکل ۸-۱۷ نشان داده شده است. در این مثال p را ۳ و q را ۱۱ فرض کرده‌ایم که نتیجه می‌دهد: $n = 33$ و $Z = 20$. یک مقدار مناسب برای d، عدد ۷ است چرا که ۲۰ و ۷ هیچ عامل مشترکی ندارند. طبق این انتخاب باید عدد e را به نحوی پیدا کرد که در رابطهٔ $7\times e \bmod 20 = 1$ صدق نماید، که عدد ۳ را نتیجه می‌دهد. کدهای رمز شدهٔ C برای یک پیام مثل P از رابطهٔ $C = P^3 \bmod 33$ بدست می‌آید. متن رمز شده را می‌توان طبق قاعده $P = C^7 \bmod 33$ از رمز خارج نمود. در این شکل برای نمونه، مراحل رمزنگاری و رمزگشایی کلمه "SUZANNE" نشان داده شده است.

به دلیل آن که اعداد اول انتخابی در این مثال بسیار کوچک است، لذا P باید کمتر از ۳۳ باشد و بدین ترتیب هر بلوک از متن فقط می‌تواند شامل یک تک کاراکتر باشد. نتیجهٔ رمز، یک جانشینی تک‌کاراکتری خواهد بود که بهیچوجه مؤثر نیست. در عوض اگر p و q را اعدادی در حدود 2^{512} انتخاب کرده بودیم، عدد n را از مرتبهٔ 2^{1024} می‌داشتیم و هر بلوک از متن می‌توانست تا ۱۰۲۴ بیت یعنی ۱۲۸ کاراکتر هشت بیتی باشد. (برخلاف بلوکهای هشت کاراکتری در DES یا بلوکهای ۱۶ کاراکتری در AES).

متن آشکار (P)			متن رمز (C)		پس از رمزگشایی	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

محاسبات فرستنده — محاسبات گیرنده

شکل ۸-۱۷. مثالی از الگوریتم RSA.

باید بدین نکته اشاره کرد که در RSA نیز شبیه به الگوریتم متقارن در حالت ECB[۱]، هرگاه ورودیهای مشابه به الگوریتم اعمال شود، نتیجهٔ خروجی یکسان خواهد بود. بدین ترتیب برای رمزنگاری داده‌ها گونه‌ای از «زنجیره‌سازی» مورد نیاز است، ولیکن در عمل اکثر سیستمهای مبتنی بر رمزنگاری کلید عمومی، از RSA فقط برای «**توزیع کلیدهای نشست**» بهره می‌گیرند تا بکمک آن کلید جدیدی تولید و ردّ وبدل گردد و پس از آن از الگوریتم متقارنی مثل «DES سه‌گانه» یا AES استفاده شود.[۲] زیرا استفاده از RSA برای رمزنگاری حجم عظیم اطلاعات، بسیار کُند عمل می‌کند و به همین دلیل عموماً از آن فقط برای توزیع کلید استفاده می‌شود.

۸ـ۳ـ۲ الگوریتمهای کلید عمومی دیگر

اگرچه از RSA به صورت گسترده‌ای استفاده می‌شود ولی این بدان معنا نیست که تنها الگوریتم شناخته شدهٔ کلید عمومی است. در حقیقت اولین الگوریتم کلید عمومی، الگوریتم Knapsack «**کوله‌پشتی**» است. (مرکِل و هِلمن؛ ۱۹۷۸) ایدهٔ این روش مبتنی بر آن است که شخصی تعداد فراوانی شیء در اختیار دارد که هر یک از لحاظ وزنی با دیگری متفاوت است. صاحب آنها، پیام خود را با انتخاب محرمانهٔ یک زیرمجموعه از این اشیاء و ریختن آنها در یک کوله‌پشتی، کدگذاری می‌کند. وزن کل اشیاء درون کوله‌پشتی و همچنین فهرست تمام اشیاء ممکن، عمومی و آشکار است در حالی که فهرست دقیق اشیاء درون کوله‌پشتی محرمانه و سرّی است. در ابتدا گمان می‌رفت که با افزودن مجموعه‌ای از محدودیتها، استخراج فهرست اشیاء درون کیسه با فرض در اختیار داشتن وزن کلّ، در عمل غیرممکن خواهد بود و می‌تواند پایهٔ یک «الگوریتم کلید عمومی» را تشکیل دهد.

مخترع این الگوریتم «رالف مرکل»، کاملاً اطمینان داشت که این الگوریتم قابل شکستن نیست و به همین دلیل اعلام کرد که به هر کس که بتواند آن را بشکند صد دلار جایزه می‌دهد. ادی شامیر (Adi Shamir یا همان S در نام RSA) سریعاً آن را شکست و جایزه را گرفت. مرکل بی‌درنگ الگوریتم را قویتر کرد و برای کسی که بتواند نسخهٔ جدید را بشکند هزار دلار پیشنهاد کرد. رونالد ری‌وست (Ronald Rivest یا همان R در RSA) سریعاً نسخهٔ جدید را نیز شکست و صاحب این جایزه شد. مرکل هرگز جرات نکرد برای نسخه جدیدتر الگوریتم خود، ده هزار دلار پیشنهاد کند و بدین نحو "A" (یعنی Leonard Adelman) سرش بی‌کلاه ماند!! الگوریتم Knapsack امن تلقی نمی‌شود و هیچگاه در عمل مورد استفاده قرار نمی‌گیرد.

---

۱. Electronic Code Book

۲. به عبارت ساده‌تر، RSA فقط در هنگام شروع یک نشست و آنهم برای مبادلهٔ یک کلید مشترک و تصادفی کاربرد دارد و پس‌از این مرحله از الگوریتم رمزنگاری متقارن استفاده می‌شود. -م

یکی دیگر از شماهای کلید عمومی، بر پیچیدگی محاسبهٔ لگاریتم گسسته (Discrete Logarithm) بنا نهاده شده است. الگوریتمهایی که از این قاعده استفاده کرده‌اند توسط El Gamal (۱۹۸۵) و Schnorr (۱۹۹۱) ابداع شده‌اند.

چند شمای دیگر نیز وجود دارد؛ مثلاً یک دسته از روشها مبتنی بر منحنی‌های بیضوی هستند (Menezes and Vanstone, 1993) ولیکن دو دسته از مهمترین الگوریتمهای کلید عمومی، براساس پیچیدگی و دشواری تجزیهٔ اعداد اول بسیار بزرگ یا محاسبهٔ لگاریتم گسستهٔ اعداد بزرگ بنا شده‌اند. این مسائل حقیقتاً لاینحل به نظر می‌رسند و ریاضی‌دانها سالها بر روی آنها کار کرده‌اند ولی هیچ راهی برای رخنه در آن نیافته‌اند.

۸ـ۴ امضاهای دیجیتالی

احراز هویت و تعیین اعتبار بسیاری از اسناد حقوقی، بازرگانی و نظائر آن، براساس وجود یا عدم وجود امضای مجاز و دستنویس در ذیل آنها، انجام می‌شود و طبعاً تصویر این اسناد ارزش قانونی ندارد. برای سیستمهای پیام‌رسان کامپیوتری که جانشین گردش کاغذ و اسناد خطی شده‌اند نیز باید روشی پیدا شود تا بتوان آنها را به گونه‌ای امضاء کرد که هرگز قابل جعل نباشد.

مسئله ابداع یک روش جایگزین به جای امضاهای دستنویس یکی از موضوعات دشوار به حساب می‌آید. در اصل به سیستمی نیاز است که براساس آن یک طرف بتواند پیامی امضاء شده را برای طرف دیگر بفرستد به گونه‌ای که شرایط زیر به درستی احراز شود:

۱. گیرنده بتواند هویت شخص فرستندهٔ پیام را بررسی کند.
۲. فرستنده بعداً نتواند محتوای پیام ارسالی خود را انکار کند.
۳. گیرنده نیز نتواند پیامهای جعلی برای خود بسازد. [و ارسال آنها را به دیگران نسبت بدهد.]

نیاز اول در سیستمهای اقتصادی و مالی حیاتی است؛ وقتی یک مشتری بانک از طریق کامپیوتر، سفارش خرید یک تُن طلا می‌دهد (!) کامپیوتر بانک باید توانایی آنرا داشته باشد تا از هویت واقعی کامپیوتر سفارش دهنده، یقین حاصل کرده و مطمئن شود که سفارش دهنده همانی است که ادعا می‌کند و باید هزینهٔ سفارش از حساب او کسر شود. به عبارت دیگر بانک باید مشتری خود را احراز هویت کند، (و در سمت مقابل، مشتری نیز از هویت بانک مطمئن شود).

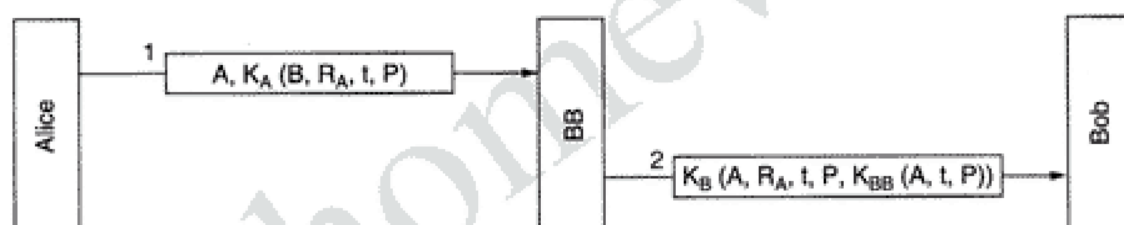
نیاز دوم بدان جهت حیاتی است که از بانک در مقابل کلاه‌برداری محافظت نماید. فرض کنید بانک، یک تن طلای سفارش داده شده را تهیه کرده ولی به ناگاه قیمت طلا بشدت سقوط می‌کند. مشتری بخت برگشته ممکن است ادعا کند که هرگز سفارش خرید یک تن طلا نداده است. وقتی بانک پیام ارسالی را به دادگاه تسلیم می‌کند، ممکن است مشتری ارسال آن را تکذیب نماید. داشتن این ویژگی که هیچیک از طرفینِ یک قراردادِ امضاء شده، نتوانند مفاد آن را تکذیب کنند اصطلاحاً "nonrepudiation" یا «**غیرقابل انکار بودن**» پیام نامیده می‌شود. الگوهای امضای دیجیتال که آنها را معرفی خواهیم کرد این قابلیت را فراهم می‌سازند.

سومین نیاز برای حفاظت از مشتری در مقابل کلاه‌برداری است مثلاً در سناریوی قبل وقتی قیمت طلا پس از خرید بشدت افزایش می‌یابد شاید بانک بخواهد پیام ارسالی مشتری را دستکاری کرده و وزن طلای سفارشی را از یک تن به یک شمش کاهش بدهد! در این سناریوی کلاه‌برداری، بانک سود باقیماندهٔ طلا را برای خود بر می‌دارد!!

۸-۱-۴ امضاهای دیجیتالی با کلید متقارن

یکی از روشهای ساماندهی امضاهای دیجیتالی آنست که یک مرکز معتبر و مجازِ گواهی امضاء داشته باشیم که همه را می‌شناسد و مورد اعتماد همه نیز هست؛ آن را اصطلاحاً BB (برادر بزرگتر یا Big Brother) می‌نامیم. هر کاربر برای خود یک کلید رمز سرّی (Secret Key) انتخاب کرده و شخصاً به ادارهٔ BB مراجعه و آن را ثبت می‌نماید. بدین ترتیب کاربری مثل آلیس، فقط خودش و BB کلید سرّی و توافق شدهٔ K_A را می‌دانند؛ بهمین روال، دیگر کاربران نیز کلید خودشان را در BB ثبت می‌نمایند.

وقتی آلیس بخواهد پیام امضاء شدهٔ خود یعنی P را برای کارپرداز بانک خود (یعنی باب) بفرستد، $K_A(B,R_A,t,P)$ را تولید می‌کند که در آن B ، مشخصهٔ شناسایی باب (Bob ID)، R_A یک عدد تصادفی که توسط آلیس انتخاب شده ، t مهر زمان برای اطمینان از جدید و تازه بودن آن، P اصل پیام و $K_A(B,R_A,t,P)$ نتیجهٔ رمزنگاری مجموعهٔ این چهار آیتم توسط کلید سرّی آلیس یعنی K_A است. سپس او این داده‌های رمز شده را طبق شکل ۸-۱۸ برای BB می‌فرستد. BB متوجه می‌شود که پیام از آلیس است لذا آن را با کلید سرّی آلیس رمزگشایی می‌کند و به نحوی که در شکل نشان داده شده آنرا مجدداً رمز کرده و برای باب می‌فرستد. پیام ارسالی به باب شامل اصل پیام آلیس و یک پیام امضاء شدهٔ $K_{BB}(A,t,P)$ است. حال باب می‌تواند درخواست آلیس را با اطمینان خاطر انجام بدهد.



شکل ۸-۱۸. امضاهای دیجیتالی به کمک مرکز گواهی امضاء (BB).

اگر بعداً آلیس ارسال پیام را انکار کرد چه اتفاقی می‌افتد؟ طبعاً هر کسی می‌تواند از دیگری ادعای خسارت کند! (حداقل در ایالات متحده این گونه است!) سرانجام وقتی مورد در دادگاه مطرح می‌شود و آلیس قویاً پیام ارسالی خود به باب را تکذیب می‌کند، قاضی از باب می‌پرسد که چگونه ادعا می‌کند پیام ارسالی آلیس حقیقتاً توسط خود آلیس ارسال شده و از شخص ثالثی مثل ترودی نیست. باب ادعا می‌کند که BB هرگز پیامی را از آلیس قبول نخواهد کرد مگر آن که با کلید شخص آلیس یعنی K_A رمز شده باشد لذا امکان آن که شخص ثالثی بتواند پیامی جعلی را از طرف آلیس بفرستد و BB آن را کشف نکند وجود نخواهد داشت.

سپس باب مدرک $K_{BB}(A,t,P)$ را به دادگاه تسلیم کرده و بیان می‌کند که این همان پیامی است که توسط BB امضاء شده است و ارسال پیام P توسط آلیس به باب را اثبات می‌کند. قاضی از BB (که مورد اعتماد همه است) می‌خواهد که این مدرک را رمزگشایی کند. وقتی BB صحت ادعای باب را تائید کرد، دادگاه به نفع باب رأی خواهد داد و پرونده مختومه خواهد شد.

یکی از مسائلی که در پروتکل امضاء در شکل ۸-۱۸ به صورت بالقوه وجود دارد آن است که ترودی ممکن است یک پیام ارسالی از آلیس را استراق سمع کرده و آن را بعداً از طرف آلیس به باب بفرستد. برای کاهش این مشکل برای هر پیام از **«مُهر زمان»** (Time Stamp) استفاده می‌شود. به علاوه، باب می‌تواند با بررسی R_A اثبات کند که آیا چنین پیامی را قبلاً نیز دریافت کرده است یا خیر. اگر پیام تکراری بود آن را حذف خواهد کرد. دقت کنید که براساس مهر زمان باب می‌تواند پیامهای قدیمی را حذف کند. برای پیشگیری از تکرار آنی و بلادرنگ پیام، باب

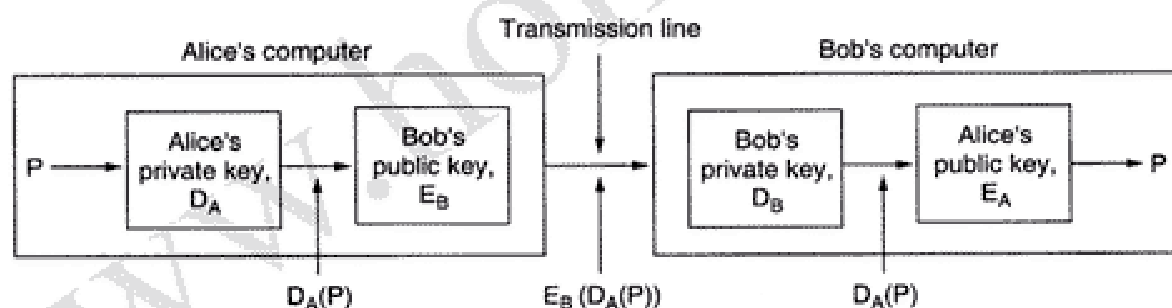
R_A را بررسی می‌کند و اگر پیامی با R_A تکراری دریافت گردد حذف می‌شود. اگر باب از این دو مورد مطمئن شد می‌تواند فرض کند که این تقاضا معتبر و جدید است.

۸-۴-۲ امضاهای باکلید عمومی

مشکل ساختاری در بکارگیری رمزنگاری با کلید متقارن برای امضاهای دیجیتالی، آن است که همه باید بـه BB (مرکز گواهی امضاء) اعتماد کنند. در ضمن BB قادر است تمام پیامهای امضاء شده را بخوانـد. منطقی‌ترین کاندیداهای راه‌اندازی مرکز گواهی امضاء ، دولت، بانکها، سازمانهای حسابرسی و کانون وکلاء هستند. متأسفانه هیچیک از این سازمانها مورد اعتماد و وفاق عموم شهروندان نیستند. بنابراین، امکان امضای مستندات به گونه‌ای که به هیچ مرکز گواهی امضاء نیاز نباشد، مورد بسیار جالبی است.

خوشبختانه، رمزنگاری باکلید عمومی می‌تواند در این زمینه نقش بسیار مؤثر و مثبتی ایفاء کند. فرض را بر آن می‌گذاریم که الگوریتمهای رمزنگاری و رمزگشایی دارای این خصوصیت است کـه $E(D(P))=P$ و هـمچنین $D(E(P))=P$.[1] (RSA دارای این ویژگی هست و چنین فرضی دور از واقعیت نخواهد بود.) با فرض وجود این ویژگی، آلیس می‌تواند متن رمز و امضا شدهٔ P را به صورت $E_B(D_A(P))$ برای باب بفرستد.[2] دقت داشته باشید که آلیس فقط و فقط خودش کلید خصوصی خود یعنی D_A را می‌داند؛ همچنین کلید عمومی باب یـعنی E_B را در اختیار دارد بنابراین ایجاد پیام فوق برای آلیس ممکن خواهد بود.

وقتی باب پیام را دریافت می‌کند، ابتدا آن را با کلید خصوصی خود رمزگشایی کرده و $D_A(P)$ را مطابق با شکل ۸–۱۹ بدست می‌آورد. او این متن را در جای امنی ذخیره می‌کند [برای استفادهٔ احتمالی در دادگاه] و سپس کلید عمومی آلیس یعنی E_A را بر روی آن اعمال کرده و متن اصلی را بدست می‌آورد.



شکل ۸–۱۹. امضاهای دیجیتالی با استفاده از رمزنگاری کلید عمومی.

برای آن که ببینیم این ساختار چگونه نیازهای امضای دیجیتالی را برآورده می‌کند، فـرض کـنید بـعداً آلیس، ارسال پیام P به باب را انکار کند. وقتی این مورد در دادگاه مطرح می‌شود، باب هر دو پیام P و $D_A(P)$ را به دادگاه تسلیم می‌کند. قاضی می‌تواند به سادگی و با اعمال کلید عمومی آلیس یعنی E_A بررسی کند که آیا باب، پیام اصلی و معتبر را دارد یا خیر! از آنجایی که باب، کلید خصوصی آلیس را در اختیار ندارد لذا نـمی‌توانسته $D_A(P)$ را بـه صورت جعلی تولید کند و قطعاً آن را به همین صورت دریافت کرده و بنابراین کسی جز آلیس آن را نـفرستاده است. آلیس در طی دوران حبس خود به جرم سوگند دروغ و کلاه‌برداری، به قدر کافی وقت دارد که به طرح یک الگوریتم جدید رمزنگاری با کلید عمومی بپردازد!!

۱. یعنی فرض شده متن رمزنگاری شده با کلید عمومی، بکمک کلید خصوصی قابل رمزگشایی است و بالعکس متن رمزنگاری شده با کلید خصوصی، بکمک کلید عمومی قابل رمزگشایی است. –م

۲. یعنی ابتدا پیام را با کلید خصوصی خودش رمز کند و نتیجه را مجدداً با کلید عمومی باب رمز و ارسال نماید. م.–

اگرچه بکارگیری رمزنگاری با کلید عمومی در طراحی امضاهای دیجیتالی، الگوی بسیار جالبی است ولی از مشکلاتی رنج می‌برد که ناشی از خود الگوریتم نیست بلکه مربوط به محیطی است که در آن عمل می‌کند. فقط زمانی باب می‌تواند ثابت کند که پیام ارسالی متعلق به آلیس است که D_A (کلید خصوصی آلیس) محرمانه و سرّی باقی بماند. اگر آلیس کلید خصوصی خود را لو بدهد، هیچ دلیل محکمه‌پسندی باقی نخواهد ماند زیرا هر کسی حتی خود باب می‌توانسته چنین پیامی را ارسال نماید.

به عنوان مثال، مشکل زمانی بروز می‌کند که باب دلال سهام آلیس باشد. آلیس به باب می‌گوید که برایش سهام یا اوراق بهادار خاصی را بخرد. بلافاصله پس از این سفارش قیمتها بشدت سقوط می‌کند. آلیس برای آن که بتواند پیام سفارش خود به باب را انکار کند، سریعاً با پلیس تماس گرفته و ادعا می‌کند که خانه‌اش مورد سرقت واقع شده و کامپیوتر محتوی کلید رمز دزدیده شده است. بسته به قوانین حاکم بر کشور یا ایالت آلیس، او ممکن است از لحاظ قانونی از خود سلب مسئولیت کند یا برعکس مسئول خسارت خود باشد، بالاخص اگر او ادعا کند موضوع سرقت خانه‌اش را به دلیل آن که سر کار بوده چندین ساعت بعد متوجه شده است!

مشکل دیگر در این ساختار آن است که اگر آلیس تصمیم بگیرد کلید رمز خود را عوض کند چه اقدامی باید انجام بدهد؟ این کار کاملاً قانونی است و حتی شاید انجام آن به صورت دوره‌ای ایدهٔ بسیار خوبی هم باشد. اگر در دادگاهی که بعداً تشکیل می‌شود، به نحوی که در بالا اشاره کردیم قاضی کلید عمومی فعلی آلیس یعنی E_A جدید را بر روی $D_A(P)$ اعمال نماید، متوجه می‌شود که P بدست نمی‌آید. باب در چنین شرایطی حسابی گیج و درمانده خواهد شد.

در اصل، از هر الگوریتم رمزنگاری با کلید عمومی می‌توان برای امضاهای دیجیتالی بهره گرفت ولیکن استاندارد غیررسمی صنعت، رمزنگاری RSA است و بسیاری از محصولات امنیتی از آن بهره گرفته‌اند. با این وجود در سال ۱۹۹۱، NIST استاندارد جدیدی برای امضاهای دیجیتال به نام DSS[۱] پیشنهاد کرد که مبتنی بر الگوریتم رمزنگاری کلید عمومی El Gamal بود. الگوریتم El Gamal امنیت ذاتی خود را از دشوار بودن محاسبهٔ لگاریتم گسسته (به جای تجزیهٔ اعداد بزرگ به عوامل اول) کسب کرده است.

طبق معمول وقتی دولت سعی در تحمیل یک استاندارد رمزنگاری می‌کند، غوغایی پیرامون آن به پا می‌شود. از DSS در موارد زیر انتقاد شده است:

۱. **بسیار محرمانه و سرّی است (NSA در این پروتکل از الگوریتم El Gamal استفاده کرده است.)**
۲. **بسیار کُند عمل می‌کند. (برای بررسی امضاء ده تا چهل برابر از RSA کُندتر عمل می‌نماید.)**
۳. **خیلی جدید است. (الگوریتم El Gamal هنوز به قدر کافی تحلیل و بررسی نشده است.)**
۴. **خیلی ناامن به نظر می‌رسد. (از کلید ۵۱۲ بیتی با طول ثابت استفاده شده است.)**

در بازبینی‌های بعدی، چهارمین مورد اشکال با افزایش طول کلید به ۱۰۲۴ بیت، برطرف شد ولیکن همچنان دو اشکال اول باقی است.

۸-۴-۳ خلاصهٔ پیامها (Message Digests)

یک انتقاد که به روش امضاء دیجیتال وارد است، آنست که اغلب دو عمل متفاوت و مجزا را درهم آمیخته و باهم انجام می‌دهند: **«احراز هویت»** و **«سرّی ماندن پیام»**. بسیاری از اوقات احراز هویت لازم است در حالی که به سرّی ماندن محتوای پیام نیازی نیست. سیستمهایی که فقط سرویس «احراز هویت» ارائه می‌کنند و در مورد مبادلهٔ سرّی

---

۱. Digital Signature Standard

پیام کاری انجام نمی‌دهند (گذشته از سرعت و کارآیی بیشتر) راحتتر مجوز صدور می‌گیرند.[۱] در زیر الگویی را برای «احراز هویت» بررسی می‌کنیم که نیازی به رمزنگاری کل پیام ندارد.

این ساختار مبتنی بر تابع Hash یک مرحله‌ای است (One-Way Hash Function) که یک قطعهٔ طولانی از متن اصلی را گرفته و یک رشتهٔ بیتی با طول ثابت از آن محاسبه و استخراج می‌کند. حاصل این **«تابع درهم‌سازی»**، اغلب به نام **«خلاصهٔ پیام»**[۲] یا MD[۳] شناخته می‌شود و دارای چهار ویژگی زیر است:

۱. با داشتن P مفروض، محاسبهٔ MD(P) بسیار ساده است.

۲. با داشتن MD(P)، عملاً پیدا کردن P غیرممکن است.

۳. با داشتن P مفروض نمی‌توان یک P' پیدا کرد به نحوی که MD(P) = MD(P')

۴. تغییر در ورودی حتی به اندازهٔ یک بیت، خروجی کاملاً متفاوتی ایجاد خواهد کرد.

برای برآورده کردن ویژگی سوم، طول رشته Hash* ، باید حداقل ۱۲۸ بیت یا ترجیحاً بیشتر باشد. برای برآورده کردن نیاز چهارم، رشته Hash باید بر خلاف الگوریتمهای رمزنگاری با کلید متقارن [که بر روی بلوکهای کوچک عمل می‌کنند] براساس تمام بیتهای درهم شدهٔ پیام محاسبه شود.

محاسبهٔ **«خلاصهٔ پیام»** براساس قطعه‌ای از یک متن، بسیار سریعتر از رمزنگاری آن توسط الگوریتمهای کلید عمومی است و بدین ترتیب محاسبهٔ MD می‌تواند برای افزایش سرعت الگوریتمهای امضای دیجیتالی مؤثر واقع شود. برای آن که ببینیم این ساختار چگونه کار می‌کند مجدداً به پروتکل امضاء در شکل ۸–۱۸ دقت کنید. به جای امضای P با ارسال $K_{BB}(A,t,P)$، مرکز گواهی امضاء (یعنی BB)، تابع MD را بر روی P اعمال کرده و یک رشتهٔ نسبتاً کوتاه MD(P) را بدست می‌آورد و با جاسازی آن در $K_{BB}(A,t,MD(P))$ [به جای $K_{BB}(A,t,P)$ که طولانی است] آنرا برای باب ارسال می‌کند.

هرگاه یک دعوای حقوقی بوجود بیاید، باب می‌تواند P و $K_{BB}(A,t,MD(P))$ را به دادگاه عرضه کند. پس از آن که مرکز گواهی آنرا برای قاضی رمزگشایی کرد، باب MD(P) را به عنوان مدرک در اختیار خواهد داشت که می‌تواند صحت پیام P را تائید کند. همچنین از آنجایی که عملاً باب قادر نخواهد بود که پیامی جعلی پیدا کند که رشتهٔ Hash آن [حاصل MD]، معادل با رشتهٔ Hash پیام واقعی باشد لذا باب نخواهد توانست دادگاه را در خصوص صحت پیام جعلی خود متقاعد کند. با استفاده از روش **«خلاصهٔ پیام»** هم در زمان رمزنگاری کل پیام و هم در هزینهٔ انتقال پیام بر روی شبکه صرفه‌جویی خواهد شد.

پیاده‌سازی روش **«خلاصهٔ پیام»** به نحوی که در شکل ۸–۲۰ نشان داده شده، براساس سیستمهای کلید عمومی نیز عملیست. در اینجا آلیس ابتدا خلاصهٔ پیام خود را محاسبه می‌نماید، سپس آن را به جای اصل پیام امضاء کرده و نهایتاً اصل پیام [رمز نشده] را به همراه MD رمز شده برای باب می‌فرستد.[۴] اگر شخص ثالثی مثل ترودی P را دستکاری کند، باب وقتی MD(P) را محاسبه می‌کند متوجه جعلی بودن پیام خواهد شد.

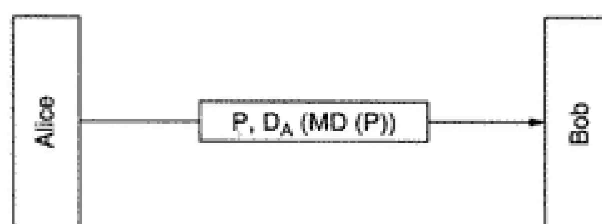
---

۱. به خاطر داشته باشید که در بسیاری از کشورها صدور نرم‌افزارهای رمزنگاری یا سیستمهای مبتنی بر مبادلهٔ سرّی پیام کلاً ممنوع است و مجازاتهای زندان (در فرانسه تا ۱۳ سال) دارد!! –م.

۲. اگر بخواهیم در دو جمله «خلاصهٔ پیام» یا همان Message Digest را بیان کنیم عبارتست از: «یک رشتهٔ رمزی کوتاه که از درون یک پیام استخراج و پس از رمزنگاری بهمراه پیام رمزنشده ارسال می‌شود. حتی اگر یک بیت از پیام اصلی دستکاری شود رشتهٔ خلاصهٔ پیام جدید با این رشته یکسان نخواهد بود.» * رشته Hash، در حقیقت همان رشته‌ای است که از درون پیام استخراج و به صورت رمز همراه آن ارسال می‌شود تا براساس آن بتوان سلامت پیام و هویت صاحب آنرا بررسی کرد. هر جا صحبت از Hash شد منظور همین رشته بیت است. –م

۳. Message Digest

۴. منظور از امضای «خلاصه پیام» یا MD، رمزنگاری آن توسط کلید خصوصی صاحب پیام می‌باشد. –م



شکل ۸-۲۰. امضاهای دیجیتالی با استفاده از «خلاصهٔ پیام».

MD5

توابع متنوعی برای تولید **«خلاصهٔ پیام»** پیشنهاد شده است که رایج‌ترین آنها MD5 (ری وست، ۱۹۹۲) و SHA-1 (NIST ، ۱۹۹۳) است. MD5 پنجمین روش پیاده سازی **«خلاصهٔ پیام»** است که همگی توسط رونالد ری‌وست (سرپرست گروه ابداع کنندهٔ RSA) طراحی شده‌اند. این روش با درهم فشردن تمام بیتها، طبق رابطه‌ای بسیار پیچیده، MD را به نحوی محاسبه می‌کند که یکایک بیتهای خروجی از یکایک بیتهای ورودی [متن اصلی] تأثیر می‌پذیرند. بطور کاملاً مجمل، این روش ابتدا آنقدر بیتهای بی‌اهمیت به متن اصلی اضافه می‌کند که طول آخرین بلوک، فقط و فقط ۴۴۸ بیت باشد. سپس طول واقعی پیام در یک فیلد ۶۴ بیتی به پیام اضافه می‌شود تا نهایتاً تعداد صحیحی از بلوکهای ۵۱۲ بیتی بدست آید. [یعنی طول کل متن اصلی ضریبی از ۵۱۲ شود.] سپس در تکمیل مراحل پیش‌پردازش، یک بافر ۱۲۸ بیتی با یک عدد ثابت، مقداردهی اولیه می‌گردد.

حال محاسبات آغاز می‌شود. در هر دور از محاسبه یک بلوک ۵۱۲ بیتی از متن ورودی استخراج و طبق یک رابطهٔ خاص با بافر ۱۲۸ بیتی ادغام [مخلوط] می‌شود. برای اندازه‌گیری بهتر، یک جدول که با مقادیر تابع Sin مقداردهی شده نیز در آن تزریق می‌شود. استفاده از تابع شناخته شده‌ای مثل Sin از آن جهت نبوده که مقادیر آن تصادفی‌تر از مولد اعداد تصادفی است بلکه فقط بدین موضوع کمک کرده که جلوی هر گونه سوءظن در خصوص طراحی یک رخنهٔ پنهان (Back door) درون الگوریتم گرفته شود. بخاطر داشته باشید که وقتی IBM از تشریح S-boxها در سیستم DES و ماهیت آنها طفره رفت، شایعات بسیار زیادی پیرامون آن در گرفت. ری‌وست (مخترع MD-5) می‌خواست از هر گونه شایعه‌ای جلوگیری کند. در ادامه، چهار دور پردازش بر روی هر بلوک ورودی انجام می‌شود. این فرآیند آنقدر تکرار می‌شود تا محاسبهٔ تمام بلوکهای ورودی خاتمه یابد. محتوای بافر ۱۲۸ بیتی ، Message Digest یا همان خلاصهٔ «پیام» را تشکیل می‌دهد.

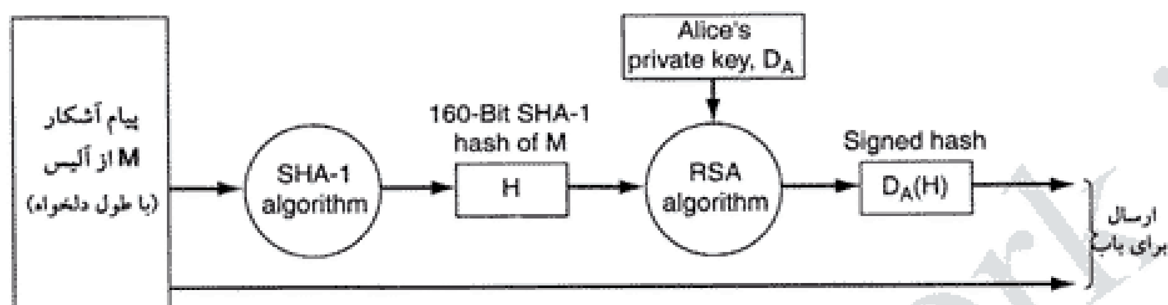
اکنون بیش از یک دهه است که MD5 معرفی شده و بسیاری از افراد سعی در حمله به آن داشته‌اند. اگرچه تعدادی نقطه ضعف در آن پیدا شده ولی برخی از مراحل درونی الگوریتم نگذاشته تا این الگوریتم درهم شکسته شود. با این وجود اگر این چند مرحلهٔ باقیمانده که حصارهای نهایی MD5 محسوب می‌گردند شکسته شود، MD5 فرو می‌پاشد ولی علیرغم این موضوع، در زمان نوشتن این کتاب، MD5 هنوز هم به طور گسترده‌ای رایج است.

SHA-1[1]

یکی دیگر از توابع مهم **«خلاصهٔ پیام»** تابع SHA-1 است که توسط NSA (آژانس امنیت ملی در آمریکا) ابداع شده و توسط NIST (ادارهٔ استانداردها و فناوریهای مدرن آمریکا) به شمارهٔ FIPS 180-1 به ثبت رسیده است. همانند MD5 ، SHA-1 نیز بلوکهای ورودی را در قالب بلوکهای ۵۱۲ بیتی پردازش می‌کند ولی برخلاف MD5 یک رشتهٔ

۱. Secure Hash Algorithm

۱۶۰ بیتی (به عنوان خلاصهٔ پیام یا Message Digest) تولید می‌کند. روش عمومی ارسال یک پیام غیرمحرمانه ولی امضاء شده از آلیس به باب در شکل ۸–۲۱ نشان داده شده است. در این شکل، پیام آلیس به الگوریتم SHA-1 تحویل می‌شود تا یک رشتهٔ «درهم شدهٔ» ۱۶۰ بیتی[۱] بدست آید. سپس آلیس این رشتهٔ ۱۶۰ بیتی را با استفاده از کلید خصوصی RSA خود رمز (امضاء) می‌کند و نهایتاً پیام اصلی رمز نشده و رشتهٔ Hash امضاء شده (رمز شده) را برای باب می‌فرستد.



شکل ۸–۲۱. استفاده از SHA-1 و RSA برای امضای پیامهای غیرمحرمانه.

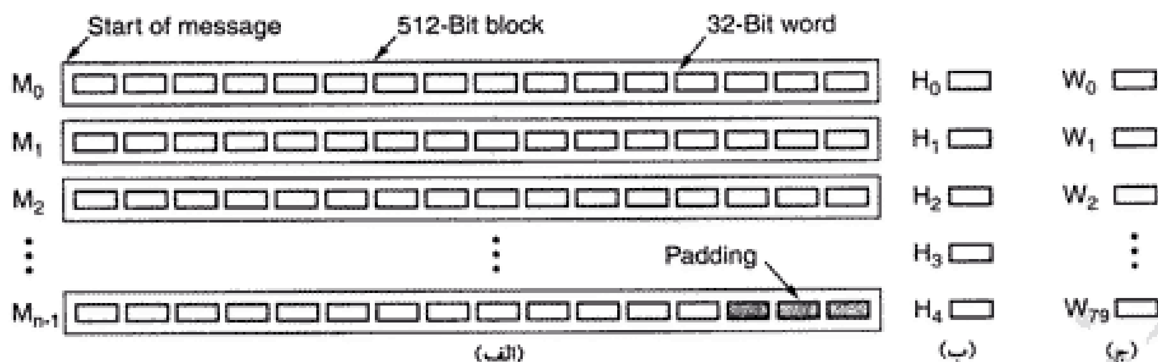
پس از دریافت، باب (بدون توجه به امضای پیام) شخصاً رشتهٔ Hash را برای پیام محاسبه می‌کند و از طرف دیگر با کلید عمومی آلیس، Hash ارسالی را بازگشایی می‌کند. اگر این دو رشته بیت با هم یکسان بودند پیام ارسالی معتبر و حقیقی است. از آنجایی که شخص ثالثی مثل ترودی قادر نیست پیام ارسالی را در حین گذر از شبکه تغییر داده و برای آن Hash جدید تولید کند [چون کلید خصوصی آلیس را نمی‌داند] لذا باب به سادگی هر گونه تغییری را که توسط ترودی در پیام ایجاد شده باشد، کشف خواهد کرد. برای پیامهایی که صحت آنها حیاتی است ولی محتوای آنها محرمانه نیست، روش شکل ۸–۲۱ کاربرد زیادی دارد. این ساختار ضمن هزینهٔ محاسباتی بسیار پایین، تضمین می‌کند که هر گونه دستکاری در پیامها حین گذر از شبکه، با احتمال بسیار بالایی کشف شود.

حال اجمالاً ببینیم که SHA-1 چگونه کار می‌کند. SHA-1 کار را با اضافه کردن یک بیت ۱ در انتهای پیام و سپس تعدادی بیت صفر در ادامهٔ آن آغاز می‌کند تا طول پیام ضریبی از ۵۱۲ بیت شود. سپس یک عدد ۶۴ بیتی که طول حقیقی پیام (قبل از اضافه کردن بیتهای صفر و یک) را نشان می‌دهد با ۶۴ بیت انتهایی OR می‌شود. در شکل ۸–۲۲، یک پیام در قالب بلوکهای ۵۱۲ بیتی به همراه بیتهای اضافه شده به آن نشان داده شده است که محل قرار گرفتن بیتهای اضافی (و فیلد ۶۴ بیتی) در انتهای پیام در سمت راست (پایین‌ترین سطر) مشاهده می‌شود. در کامپیوترها این ساختار شبیه به ماشینهای Big-Endian[۲] مثل SPARC است ولی فارغ از نوع ماشین، SHA-1 داده‌های اضافی (Pad) را به انتهای پیام می‌چسباند. در خلال محاسبه، SHA پنج متغیر ۳۲ بیتی H_0 تا H_4 را در حافظه نگهداری می‌کند که رشتهٔ Hash [۱۶۰ بیتی] در آنجا جمع‌آوری و محاسبه می‌شود. در ادامه بلوکهای ۵۱۲ بیتی M_0 تا M_{n-1} به ترتیب پردازش می‌شوند. برای بلوک جاری ابتدا ۱۶ کلمهٔ ۳۲ بیتی [معادل ۵۱۲ بیت] به نحوی که در شکل ۸–۲۲–ج نشان داده شده در یک آرایه کمکی ۸۰ کلمه‌ای به نام W کپی می‌شود. ۶۴ کلمهٔ باقیمانده از آرایهٔ W طبق فرمول زیر محاسبه و پر می‌شوند:

$$W_i = S^1 (W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \qquad (16 <= i =< 79)$$

۱. 160 Bits SHA-1 Hash

۲. ماشینهای Big Endian ماشینهایی هستند که برای ذخیره سازی کلمات ۲، ۴ یا ۸ بایتی در حافظه ابتدا بایت پرارزش را و سپس بایتهای کم ارزش را ذخیره می‌کنند. ماشینهای سری x86 برعکس عمل می‌کنند.



شکل ۸-۲۲. (الف) به پیام آنقدر داده‌های زائد اضافه می‌شود تا طول آن ضریبی از ۵۱۲ شود. (ب) متغیرهای خروجی. (ج) آرایه‌ای از کلمات ۴ بایتی.

در فرمول فوق، $S^b(W)$ به معنای شیفت چرخشی به سمت چپ در کلمهٔ ۳۲ بیتی W و به اندازهٔ b بیت است. حال پنج متغیر جدید A تا E به ترتیب با مقادیر H_0 تا H_4 مقداردهی می‌شوند.

محاسباتی را که در ادامه بر روی A تا E انجام می‌شود می‌توان به صورت شبه‌کد C نشان داد:

```
for (i = 0 ; i<80 ; i ++) {
    temp = S^5(A)+f_i(B,C,D)+E+W_i+K_i;
    E=D;  D=C  C=S^30(B);  B=A;  A=temp;
}
```

در کُد بالا K_iها ثابتهایی هستند که در استاندارد تعریف شده‌اند. تابع مخلوط‌سازی[1] f_i نیز به ترتیب زیر تعریف شده است:

$f_i(B,C,D)=(B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D)$ $(0 \le i \le 19)$

$f_i(B,C,D)=B \text{ XOR } C \text{ XOR } D$ $(20 \le i \le 39)$

$f_i(B,C,D)=(B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D)$ $(40 \le i \le 59)$

$f_i(B,C,D)=B \text{ XOR } C \text{ XOR } D$ $(60 \le i \le 79)$

پس از آنکه حلقه هشتاد بار اجرا و تکمیل شد، متغیرهای A تا E به ترتیب با H_0 تا H_4 جمع می‌شوند.

در اینجا اولین بلوک ۵۱۲ بیتی پردازش شده است و همین روال برای بلوک بعدی آغاز و تکرار می‌شود. آرایهٔ W با بلوک جدید مقداردهی می‌گردد در حالی که مقادیر H_0 تا H_4، از مرحلهٔ قبلی [بدون تغییر] حفظ خواهد شد. هر گاه این بلوک نیز پردازش شد، بلوکهای بعدی نیز به همین ترتیب پردازش می‌شوند تا تمام بلوکهای ۵۱۲ بیتی در این ملغمه وارد شوند! وقتی آخرین بلوک پردازش شد، محتوای پنج کلمهٔ ۳۲ بیتی H_0 تا H_4 در آرایهٔ H به عنوان رشتهٔ ۱۶۰ بیتی Hash بدست می‌آید. برنامهٔ کامل SHA-1 به زبان C در RFC 3174 ارائه شده است.

نسخهٔ جدید SHA-1 با رشته‌های ۲۵۶، ۳۸۴ و ۵۱۲ بیتی Hash تحت مطالعه و پیاده‌سازی است.

۸-۴-۴ حمله روز تولد (The birthday Attack)

در دنیای رمزنگاری هیچ چیزی مثل آنچه در ظاهر به نظر می‌رسد، نیست! شاید کسی فکر کند برای شکستن Message Digest به طول m بیت، باید 2^m حالت مختلف یکی‌یکی آزمایش شود. ولی در حقیقت با استفاده از مفهوم **«حملهٔ روز تولد»** تعداد عملیات آزمایش $2^{m/2}$ حالت است؛ روشی که توسط Yuval (۱۹۷۹) در مقاله‌ای به

---

۱. Mixing Function

نام "How to Swindle Rabin" منتشر شد. ایدهٔ این نوع حمله از تکنیکی منشاء می‌گیرد که اساتید ریاضی در کلاس درس احتمال، بکار می‌برند. سؤال این است که: «در یک کلاس چند نفر باید حضور داشته باشند تا احتمال آن که دو نفر در این جمع در یک روز متولد شده باشند بیش از $\frac{۱}{۲}$ باشد؟» بسیاری از دانشجویان انتظار دارند جواب این مسئله بیش از ۱۰۰ باشد، ولیکن تئوری احتمال می‌گوید که این تعداد ۲۳ است. بدون یک تحلیل دقیق و به صورت ذهنی می‌توان بدین گونه حساب کرد که با ۲۳ نفر ۲۵۳=$\frac{۲۳\times۲۲}{۲}$ زوج مختلف خواهیم داشت که هر یک از این زوجها با احتمال $\frac{۱}{۳۶۵}$ در یک روز سال به دنیا آمده‌اند. [بنابراین احتمال آن که یک زوج در یک روز به دنیا آمده باشد $\frac{۲۵۳}{۳۶۵}$ و بیش از ۰/۵ است] که با این استدلال عدد ۲۳ چندان عجیب نیست.

بطور عام اگر یک نگاشت بین ورودی و خروجی، با n ورودی (مردم، پیامها و نظائر آن) و k حالت مختلف خروجی (تولد، Message Digest و نظایر آن) انجام شود، $n.(n-1)/2$ زوج ورودی خواهیم داشت. اگر $n(n-1)/2>k$ باشد احتمال آن که حداقل یک مورد تطابق حاصل شود، بسیار بالا خواهد بود. بنابراین اگر $n>\sqrt{k}$ باشد احتمال آن که حداقل یک مورد تطابق پیدا شود [مثلاً دو مورد تولد در یک روز، یا تطابق پیام با Message Digest] بالاست. نتیجهٔ فوق بدان معناست که شکستن[1] یک خلاصهٔ پیام (Message Digest) ۶۴ بیتی با تولید $۲^{۳۲}$ پیام مختلف و مقایسهٔ آنها امکان‌پذیر است. [حدود چهار میلیارد حالت مختلف]

بگذارید یک مثال عملی را بررسی کنیم. دانشکدهٔ علوم کامپیوتر در یک دانشگاه به یک عضو هیئت علمی نیاز دارد که برای این موقعیت دو نفر به نامهای «تام» و «دیک» نامزد شده‌اند. تام دو سال زودتر استخدام شده و به همین دلیل زودتر از دیک برای مصاحبه دعوت می‌شود و در این صورت «دیک» این موقعیت را از دست خواهد داد. تام می‌داند که «مرلین»، مدیر گروه دانشکده، نظر مساعدی به کار او دارد، به همین دلیل از او خواهش می‌کند یک توصیه‌نامه برای او خطاب به «دین» (Dean) مسئول گزینش تام، بنویسد. تمام نامه‌ها محرمانه خواهد ماند.

«مرلین» از منشی خود «الن» می‌خواهد تا طی نامه‌ای خطاب به «دین» آنچه را مورد نظر اوست بنویسد تا پس از حاضر شدن، آنرا بررسی کرده و از طریق یک رشتهٔ ۶۴ بیتی Message Digest امضاء نماید. پس از آن «الن» می‌تواند نامه را از طریق پست الکترونیکی ارسال کند. از بخت بدِ تام، «الن» متمایل به انتخاب «دیک» است لذا ابتدا نامهٔ زیر را با ۳۲ گزینهٔ مختلف می‌نویسد (با مضمون خوب).

Dear Dean Smith,

This *[letter* | *message]* is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* | *worked with*] Prof. Wilson for [*about* | *almost*] six years. He is an [*outstanding* | *excellent*] researcher of great [*talent* | *ability*] known [*worldwide* | *internationally*] for his [*brilliant* | *creative*] insights into [*many* | *a wide variety of*] [*difficult* | *challenging*] problems.

He is also a [*highly* | *greatly*] [*respected* | *admired*] [*teacher* | *educator*]. His students give his [*classes* | *courses*] [*rave* | *spectacular*] reviews. He is [*our* | *the Department's*][*most popular* | *best-loved*] [*teacher* | *instructor*].

[*In addition* | *Additionally*] Prof. Wilson is a [*gifted* | *effective*] fund raiser. His [*grants* | *contracts*] have brought a [*large* | *substantial*] amount of money into [*the* | *our*] Department. [*This money has* | *These funds have*] [*enabled* | *permitted*] us to [*pursue* | *carry out*] many [*special* | *important*] programs, [*such as* | *for example*] your State 2000 program. Without these

---

۱. شکستن خلاصهٔ پیام به معنای آنست که: رمزشکن یک رشتهٔ خلاصهٔ پیام (Message Digest) داشته باشد و بتواند یک پیام معادل با آن پیدا کند. —م.

funds we would [*be unable* | *not be able*] to continue this program, which is so [*important* | *essential*] to both of us. I strongly urge you to grant him tenure.

پس از نگارش و تایپ نامهٔ فوق، نامهٔ دومی (با مضمون بد) به صورت زیر می‌نویسد:

Dear Dean Smith,

This [*letter* | *message*] is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* | *worked with*] Tom for [*about* | *almost*] six years. He is a [*poor* | *weak*] researcher not well known in his [*field* | *area*]. His research [*hardly ever* | *rarely*] shows [*insight in* | *understanding of*] the [*key* | *major*] problem of [*the* | *our*] day.

Furthermore, he is not [*respected* | *admired*] [*teacher* | *educator*]. His students give his [*classes* | *courses*] [*poor* | *bad*] reviews. He is [*our* | *the Department's*] least popular [*teacher* | *instructor*], known [*mostly* | *primarily*] within [*the* | *our*] Department for his [*tendency* | *propensity*] to [*ridicule* | *embarrass*] students [*foolish* | *imprudent*] enough to ask questions in his classes.

[*In addition* | *Additionally*] Tom is a [*poor* | *marginal*] fund raiser. His [*grants* | *contracts*] have brought only a [*meager* | *insignificant*] amount of money into [*the* | *our*] Department. Unless new [*money is* | *funds are*] quickly located, we may have to cancel some essential programs, such as your State 2000 program. Unfotunately, under these [*conditions* | *circumstances*] I cannot in good [*conscience* | *faith*] recommend him to you for [*tenure* | *a permanent position*].

حال «الن» کامپیوتر خود را به نحوی برنامه‌ریزی می‌کند تا در خلال شب تمام 2^{32} حالت مختلف «خلاصهٔ پیام» (Message Digest) را برای هر دو نامه محاسبه کند.[1] زمانی او موفق است که دو پیام فوق دارای «خلاصهٔ پیام» یکسان شوند. اگر نشد او چند گزینهٔ دیگر به پیام دوم می‌افزاید و در پایان هفته مجدداً آنها را می‌آزماید. فرض کنید که بالاخره یک مورد تطابق پیدا می‌کند. نامهٔ خوب را A و نامهٔ بد را B بنامید.

«الن» نامهٔ A را جهت تائید برای «مرلین» می‌فرستد و نامهٔ B را به صورت محرمانه نزد خود نگاه می‌دارد و آنرا به هیچ کس نشان نمی‌دهد. «مرلین» محتوای نامهٔ A را تائید و آنرا از طریق یک «خلاصهٔ پیام» شصت و چهار بیتی امضاء می‌کند. «الن» مستقلاً به جای نامهٔ اول، نامهٔ دوم (B) را می‌فرستد، در حالی که امضای دو نامه یکی است.

پس از دریافت نامه و «خلاصهٔ پیام» امضاء شدهٔ آن، «دین» (مسئول گزینش) الگوریتم MD را بر روی نامهٔ دوم اعمال کرده و می‌بیند که با آنچه که «مرلین» امضاء کرده مطابقت دارد فلذا [براساس مضمون نامه] «تام» را بیرون می‌اندازد. «دین» متوجه نخواهد شد که «الن» به گونه‌ای برنامه‌ریزی کرده که دو نامه با MD مشابه ایجاد شود و نامهٔ جعلی دوم را ارسال نموده است. (پایان اختیاری داستان: «الن» موضوع را به «دیک» می‌گوید. «دیک» را ترس فرا می‌گیرد و دوستی خود را با او بهم می‌زند. «الن» بشدت ناراحت می‌شود و نزد «مرلین» اعتراف می‌کند. «مرلین» موضوع را به «دین» اطلاع می‌دهد. «تام» منصب مورد نظر را بدست می‌آورد!!!!!)

اِعمال «**حملهٔ روز تولد**» بر علیه MD5 بسیار مشکل است و حتی اگر در ثانیه یک میلیارد Message Digest متفاوت تولید شود محاسبهٔ MD برای 2^{64} حالت مختلف برای دو نامه حدود ۵۰۰ سال طول می‌کشد و ضمناً

---

۱. ترکیبات مختلف کلماتی که در متن بصورت ایتالیک نمایش داده شده‌اند در مجموع حدود 2^{32} حالت خواهد بود. -م.

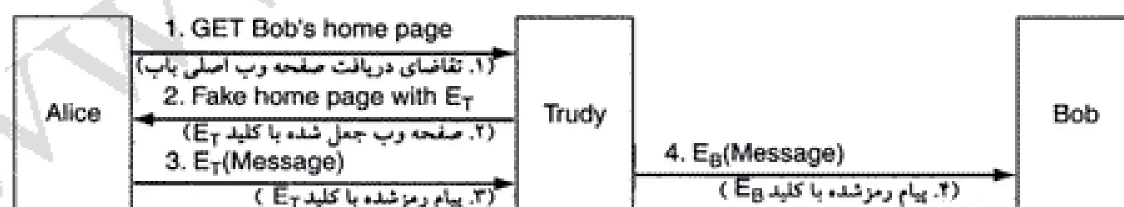
رسیدن به MD مشابه برای دو نامه تضمین شده هم نیست. البته با داشتن ۵۰۰۰ کامپیوتر که بطور موازی با هم کار کنند این زمان به پنج هفته کاهش خواهد یافت. SHA-1 در این مورد بهتر است چرا که رشته Message Digest طولانی‌تری ایجاد می‌کند.

۸ـ۵ مدیریت کلیدهای عمومی

«رمزنگاری با کلید عمومی» این امکان را فراهم آورده تا افرادی که از قبل بر روی یک کلید مشترک توافق نداشته‌اند، بتوانند با یکدیگر محاوره و مبادلهٔ اطلاعات داشته باشند. همچنین امضای پیامها را بدون نیاز به حضور یک شخص ثالث و مورد اعتماد، ممکن کرده است. نهایتاً، امضای پیامها با Message Digest ، امکان بررسی صحّت پیامهای دریافتی را میسر ساخته است.[1]

با این حال، مشکلی وجود دارد که نگاهی اجمالی بدان خواهیم انداخت: اگر آلیس و باب شناختی از هم نداشته باشند چگونه کلید عمومی یکدیگر را بدست بیاورند تا مبادلهٔ اطلاعات را آغاز کنند؟ ساده‌ترین راه حل (یعنی قرار دادن کلید عمومی در وب‌سایت شخصی خود) به دلیل زیر کارآمد نخواهد بود: فرض کنید که آلیس می‌خواهد کلید عمومی باب را بر روی وب‌سایت او نگاه کند. چه کاری انجام می‌دهد؟ او URL وب‌سایت را تایپ می‌کند. مرورگر (Browser) او از DNS برای پیدا کردن آدرس سایت شخصی باب کمک گرفته و مطابق شکل ۸-۲۳ برای دریافت این صفحهٔ وب، فرمان GET [از فرامین HTTP] را صادر می‌کند. متأسفانه ترودی در حال استراق سمع این تقاضا است و در پاسخ به آن سریعاً یک صفحهٔ وب جعلی بر می‌گرداند که احتمالاً کپی صفحهٔ وب باب است با این تفاوت که کلید عمومی باب با کلید عمومی ترودی عوض شده است. [این کلید جعلی را E_T فرض کنید.] وقتی آلیس اولین پیام خود را با E_T رمز می‌کند و به گمان خود آن را برای باب می‌فرستد ترودی آن را گرفته و با کلید خصوصی خود رمزگشایی می‌کند و پس از بهره‌برداری، برای آن که این موضوع کشف نشود پیام را با کلید واقعی باب مجدداً رمز کرده و برای او می‌فرستد؛ بدون آن که باب آگاه باشد که پیام دریافتی او استراق سمع شده است. از آن بدتر، ترودی قادر خواهد بود که پیام آلیس را قبل از رمزنگاری مجدد، تغییر نیز بدهد.

به وضوح برای پیشگیری از چنین حملاتی به مکانیزمهایی نیاز است تا مطمئن شویم که کلیدها عمومی به صورت مطمئن مبادله می‌شوند.



شکل ۸-۲۳. روشی که بر اساس آن ترودی می‌تواند رمزنگاری کلید عمومی را با اشکال مواجه کند.

۸ـ۵ـ۱ گواهینامه‌ها (Certificates)

در اولین تلاش برای توزیع مطمئن کلیدهای عمومی، می‌توانیم به یک سازمان مرکزی برای توزیع کلیدها بیندیشیم که به صورت شبانه‌روزی فعال است و کلیدهای عمومی افراد را در اختیار قرار می‌دهد. یکی از مشکلات بی‌شمار این روش آنست که چندان قابل توسعه نیست و مرکز توزیع کلید سریعاً به یک گلوگاه در شبکه تبدیل می‌شود. [زیرا زیر بار تقاضاها خواهد ماند.] همچنین اگر زمانی این مرکز از کار بیفتد، امنیت اینترنت نیز به ناگاه مختل

---

۱. به خاطر داشته باشید که Message Digest نیز توسط کلید عمومی فرد امضا کنندهٔ پیام، از رمز خارج می‌شود. -م.

خواهد شد.

به دلائل فوق، عموم افراد راهکار متفاوتی اتخاذ کرده‌اند که در آن نیازی به وجود یک مرکز فعال و تمام وقت نیست. در حقیقت، اجباری برای روی‌خط (on-line) نگاه داشتن چنین مرکزی وجود ندارد. در عوض، تنها کاری که این مرکز باید انجام بدهد آن است که کلید عمومی متعلق به افراد، شرکتها و سازمانها را «**گواهی**» کند. امروزه سازمانی که کلیدهای عمومی افراد را گواهی می‌کند اصطلاحاً **CA** (Certification Authority) نامیده می‌شود.

به عنوان مثال فرض کنید باب می‌خواهد به آلیس و دیگران اجازه بدهد تا به صورت محرمانه با او مبادلهٔ داده داشته باشند. او می‌تواند با در دست داشتن پاسپورت یا گواهینامهٔ رانندگی و همچنین کلید عمومی خود به مرکز CA مراجعه کرده و از آنها بخواهد که کلید عمومی او را گواهی کنند. CA برای او یک گواهینامه مشابه با شکل ۸-۲۴ صادر کرده و «رشتهٔ SHA-1 Hash»[1] این گواهینامه را استخراج و آنرا با کلید خصوصی خودش امضاء (رمز) می‌کند. سپس باب هزینهٔ مورد نظر CA را پرداخته و یک دیسک نرم (فلاپی) حاوی گواهینامه و رشتهٔ Hash امضاء شده را تحویل می‌گیرد.

I hereby certify that the public key 19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A belongs to Robert John Smith 12345 University Avenue Berkeley, CA 94702 Birthday: July 4, 1958 Email: bob@superdupernet.com
SHA-1 hash of the above certificate signed with the CA's private key

شکل ۸-۲۴. گونه‌ای از یک گواهینامه و رشتهٔ Hash امضاء شدهٔ آن.

کار اصلی یک گواهینامه آنست که نام صاحب کلید (نام شخصی، نام شرکت یا نظایر آن) را به همراه کلید عمومی او قید کرده باشد. باب ممکن است تصمیم بگیرد که گواهینامهٔ جدید خودش را در وب‌سایت شخصی خود قرار داده و در صفحهٔ اصلی یک لینک با این مضمون تعبیه کند: «برای دریافت گواهی دیجیتالی من اینجا کلیک کنید!» با کلیک کردن، گواهینامهٔ دیجیتالی و «**بلوک امضای** CA» (یعنی رشتهٔ Hash گواهینامه که توسط کلید خصوصی CA رمز شده است) برای متقاضی ارسال می‌شود.

حال مجدداً سناریوی شکل ۸ـ۲۳ را برای روش جدید به اجراء می‌گذاریم. وقتی ترودی تقاضای دریافت صفحهٔ وب‌سایت باب را استراق سمع می‌کند، چه کاری می‌تواند انجام بدهد؟ ترودی می‌تواند بر روی صفحهٔ وب جعلی (که بدروغ برای آلیس می‌فرستد) گواهینامهٔ خودش را ارسال کند در حالی که وقتی آلیس این گواهینامه را مطالعه می‌کند متوجه می‌شود که در حال صحبت با باب نیست زیرا نام باب را در گواهینامهٔ ارسالی مشاهده نمی‌کند. همچنین ترودی می‌تواند صفحهٔ وب متعلق به باب را در حین ارسال دستکاری کرده و کلید عمومی او را با کلید عمومی خودش عوض کند ولیکن، وقتی آلیس الگوریتم SHA-1 را بر روی این کلید عمومی اجرا می‌کند، رشتهٔ Hash بدست آمده با رشتهٔ Hash که از رمزگشایی امضاء (با کلید عمومی و شناخته شدهٔ CA) حاصل می‌شود مطابقت ندارد. از آنجایی که ترودی کلید خصوصی CA را در اختیار ندارد لذا به هیچ‌وجه قادر نخواهد

---

۱. یعنی خلاصهٔ پیام استخراج شده بروش SHA-1

بود که بلوک امضای دستکاری شده در صفحهٔ وب را رمز کند. بدین ترتیب آلیس می‌تواند مطمئن باشد که کلید عمومی باب را در اختیار دارد نه کلید ترودی یا کس دیگری را. بنابراین همانگونه که قبلاً قول داده بودیم، در این ساختار لازم نیست که CA برای بررسی گواهینامه همیشه فعال و روی خط (Online) باشد و بدین ترتیب خطر بروز گلوگاه رفع خواهد شد.

هرچند عملکرد اصلی و استاندارد گواهینامهٔ دیجیتالی آنست که کلید عمومی شخص را با مشخصات صاحب اصلی آن قید کند، می‌توان از آن بدین نحو نیز استفاده کرد که کلید عمومی بهمراه **«ویژگی»** (Attribute) صاحب آن قید شود. به عنوان مثال یک گواهینامهٔ دیجیتالی می‌تواند بیان کند که: «این کلید عمومی به شخصی با سن بالای ۱۸ سال تعلق دارد!» این گواهینامه می‌تواند [بدون فاش کردن نام و مشخصات شخص] ثابت کند که صاحب این کلید خردسال نیست و مجاز به استفاده از مفادی است که مناسب بچه‌ها نیستند. عموماً کسی که صاحب چنین گواهینامه‌ای است آن را برای سایتهای وب، اشخاص یا پروسه‌ها می‌فرستد تا در خصوص رده سنی او مطمئن شوند. این سایتها ، اشخاص یا پروسه‌ها یک عدد تصادفی تولید و آن را با کلید عمومی موجود در گواهینامه طرف مقابل رمز کرده و برای او می‌فرستند. در صورتی که صاحب گواهینامه توانست آن را رمزگشایی کرده و پس بفرستد، ثابت می‌شود که او همان کسی است که ویژگیهایش در گواهینامه درج شده است. به جای این کار می‌توان از همین عدد تصادفی برای تولید یک کلید نشست جدید و محاورهٔ مطمئن (رمزنگاری‌شده) بهره گرفت.

یک مثال دیگر که در آن به «گواهینامهٔ حاوی ویژگی» (attribute) نیاز خواهد بود سیستمهای شیءگرای توزیع‌شده هستند. هر «شیء» عموماً دارای چندین «متود» است. مالک شیء می‌تواند برای هر مشتری یک گواهینامهٔ دیجیتالی فراهم کند که در آن یک رشته بیتی درج شده و مشخص می‌کند آن مشتری، مجاز به فراخوانی چه متودهایی از آن «شیء» است و سپس این رشتهٔ بیت را به همراه کلید عمومی مشتری امضاء و ضمیمه می‌کند. بار دیگر، اگر کسی که گواهینامه دارد بتواند ثابت کند که کلید خصوصی خود را در اختیار دارد، اجازه خواهد داشت متودهایی که در این رشتهٔ بیت مشخص شده را، بکار بگیرد (بعبارت دیگر فراخوانی و اجراء کند). روش فوق این ویژگی را دارد که لازم نیست مشخصات واقعی مالک گواهینامه مشخص باشد؛ این خصوصیت برای حفظ حریم خصوصی افراد، بسیار مفید است.

۸-۵-۲ X.509

اگر هر کسی که می‌خواهد چیزی را امضا کند به CA مراجعه کند و نوع متفاوتی از گواهینامه بگیرد، چیزی نخواهد گذشت که مدیریت اشکال گوناگون و غیراستاندارد گواهینامه‌ها به یک مشکل عمده تبدیل خواهد شد. برای حل این مشکل، استانداردی برای صدور گواهینامه‌های دیجیتالی ابداع و توسط ITU تأئید شده است. این استاندارد X.509 نامیده می‌شود و امروزه در اینترنت کاربرد گسترده‌ای دارد. پس از استانداردسازی اولیه در ۱۹۸۸، تاکنون سه نسخهٔ متفاوت از X.509 ارائه شده است.

X.509 بشدت تحت تأثیر فضای حاکم بر OSI بوده است و برخی از بدترین ویژگیهای آن (مثل قواعد نامگذاری و روش کُد کردن) را به عاریه گرفته است!! خوشبختانه، IETF با X.509 همراهی کرد، کما اینکه تقریباً در تمام زمینه‌ها از آدرس‌دهی ماشینها و پروتکلهای لایهٔ انتقال گرفته تا قالب نامه‌های الکترونیکی، دخالت و همراهی کرده و عموماً در اغلب آنها از OSI[۱] چشمپوشی نموده و فقط سعی کرده کار را به درستی انجام بدهد. (فارغ از پیچیدگیهای دست و پاگیری که سازمان جهانی استاندارد همیشه تحمیل می‌کند.)

X.509 در اصل روشی برای تعریف و تبیین گواهینامه‌های دیجیتالی است. فیلدهای اصلی یک گواهینامهٔ

---

۱. استاندارد جهانی ISO برای شبکه‌های باز

X.509 در شکل ۸-۲۵ فهرست شده‌اند. توضیحاتی که در این جدول وجود دارد می‌تواند کاربرد کلّی هر فیلد را مشخص کند. برای اطلاعات بیشتر لطفاً از RFC 2459 راهنمایی بگیرید.

به عنوان مثال اگر باب در قسمت «وام» از «بانکِ پول» (Money Bank) مشغول به کار باشد، ممکن است آدرس X.500[۱] او برای درج در گواهینامه به صورت زیر باشد:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

که در آن C مشخصهٔ کشور، O مشخصهٔ سازمان (Organization)، OU مشخصهٔ قسمت (بخش در سازمان) و CN مشخصه نام عمومی فرد است. دیگر مشخصات گواهینامه نیز به روش مشابه نامگذاری می‌شوند.

یکی از مشکلات ذاتی در نامگذاری X.500 آن است که اگر آلیس تلاش کند تا با کسی به آدرس bob@moneybank.com و دارای گواهینامهٔ X.500 ، ارتباط برقرار کند، درگواهینامه‌ای که مشاهده می‌کند نام باب به وضوح دیده نمی‌شود. خوشبختانه در نسخهٔ سوم از استاندارد X.509، اجازه داده شده که نامهای DNS (اسامی و آدرسهای نمادین در اینترنت) به جای اسامی X.500 در گواهینامه‌ها درج شود که بدین نحو مشکل فوق حل شده است.

گواهینامه‌های دیجیتالی پس از تدوین، به روش OSI ASN.1[۲] کدگذاری می‌شود که می‌توانید فکر کنید تعریف و ساختاری شبیه به استراکچر در زبان C دارد، با این تفاوت که از نمادهای عجیب و غریب و بسیار طولانی استفاده شده است. اطلاعات بیشتر در مورد X.509 در (Ford and Baum, 2000) در دسترس می‌باشد.

نام فیلد	کاربرد
Version	نسخه استاندارد X.509 را مشخص می‌کند.
Serial number	این شماره بهمراه نام CA هویت (و اصالت) این گواهینامه را بصورت یکتا مشخص می‌کند.
Signature algorithm	الگوریتم بکاررفته برای امضای گواهینامه را مشخص می‌کند.
Issuer	نام X.509 برای CA (نام مرکز گواهی امضا)
Validity period	زمان شروع و ختم اعتبار گواهینامه
Subject name	موجودیتی که کلید عمومی او در این گواهینامه تایید می‌شود. (مثل شخص یا موسسه)
Public key	کلید عمومی متعلق به صاحب گواهنامه و مشخصه الگوریتم مورد استفاده او
Issuer ID	یک شماره‌شناسایی منحصر بفرد و یکتا که هویت صادرکننده گواهنامه را تعیین می‌کند. (اختیاری)
Subject ID	یک شماره‌شناسایی منحصر بفرد و یکتا که هویت صاحب گواهینامه را تعیین می‌کند. (اختیاری)
Extensions	بکمک این فیلد می‌تواند بهر تعداد مشخصات اضافی تعریف کرد.
Signature	امضای کل گواهینامه (امضا شده با کلید خصوصی CA)

شکل ۸-۲۵. فیلدهای اصلی در گواهینامهٔ X.509.

۸-۵-۳ زیرساخت کلید عمومی (Public Key Infrastructure)

به وضوح، داشتن یک مرکز مجاز صدور گواهی (CA) که تمام گواهینامه‌ها را در دنیا صادر کند، بزودی از کارآیی ساقط خواهد شد؛ زیرا در زیر بار زیاد فرو می‌پاشد و در نوع خود یک «**نقطهٔ حساس به خرابی**» محسوب می‌شود. یک راهکار دیگر آن است که چندین CA (مرکز گواهی) داشته باشیم که هر یک توسط یک سازمان خاص راه‌اندازی شده‌اند ولی همه از یک «کلید خصوصی» (Private Key) مشابه برای امضای گواهینامه‌ها استفاده

---

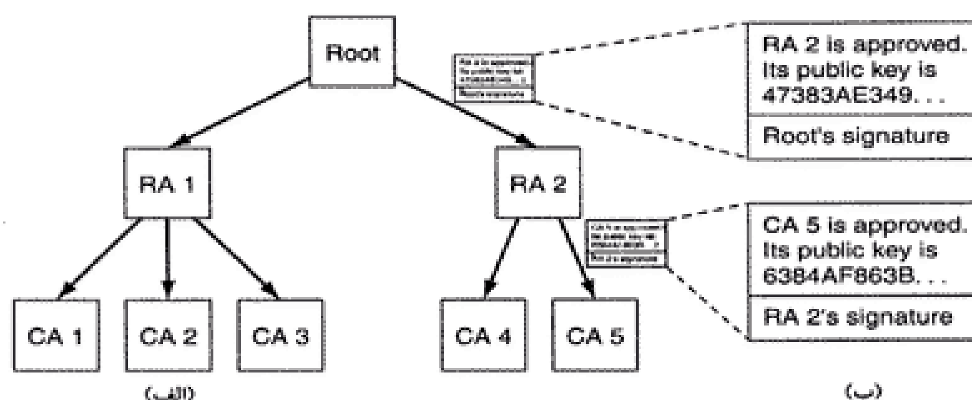
۱. X.500 استاندارد نامگذاری و X.509 استاندارد گواهینامه‌های دیجیتالی است.

۲. Abstract Syntax Notation 1

کنند. این راهکار اگرچه مسئله بار و مشکل خرابی را رفع می‌کند ولیکن مشکل جدیدی را بوجود خواهد آورد: **«مشکل نشت کردن و فاش شدن کلید»**. اگر دهها سرویس‌دهندهٔ پراکنده در سطح دنیا وجود داشته باشد و همه از یک کلید خصوصی مشابه استفاده کنند احتمال سرقت کلید یا لو رفتن آن بشدت افزایش می‌یابد. از یک طرف توافق بر سر کلید خصوصی مشابه، زیربنای امنیت دنیای الکترونیک را متزلزل می‌کند و از طرف دیگر داشتن یک سازمان مرکزی صدورگواهینامه (CA) بسیار پرمخاطره است. به علاوه، چه سازمان یا نهادی باید متولی راه‌اندازی CA شود؟ به سختی بتوان به یک مرکز مجاز جهانی اندیشید که بتواند در سطح کل دنیا مورد وفاق و اعتماد عمومی قرار بگیرد. در بعضی از کشورها، مردم اصرار دارند که این مرکز به دست دولت اداره شود در حالی که در برخی دیگر کشورها، اصرار عمومی آن است که دولت در این خصوص دخالت نکند!

بدین دلائل روشهای متفاوتی برای گواهی کلیدهای عمومی افراد معرفی شده است. این روشها همگی تحت عنوان PKI (Public Key Infrastructure) مشهور هستند. در این بخش مبانی عمومی PKI را بـه اختصار بررسی می‌کنیم، هر چند در این خصوص پیشنهادات بسیار زیاد است و جزئیات آن هنوز در حال تکوین و تدوین می‌باشد.

PKI از چندین مولفه شامل **«کاربران»**، **«CAها[1] یا مراکز صدور گواهی»**، **«گـواهـینامه‌ها»** و **«دایـرکتوریها»** تشکیل شده است. آنچه که PKI باید انجام دهد آن است که تمام این مولفه‌ها را تحت لوای یک ساختار واحـد جمع کند و برای مستندات و پروتکلهای مختلفی که در این خصوص عرضه شده، استاندارد مدونی تعریف نماید. یک الگوی بسیار ساده از PKI، به نحوی که در شکل ۸-۲۶ نشان داده شده، ساختار سلسله مراتبی است. در این مثال سه سطح را نشان داده‌ایم که در عمل می‌تواند بیشتر یا کمتر باشد. بالاترین سطح CA یعنی «ریشه» (Root)، فقط مراکز صدور گواهینامهٔ سطح ۲ را تائید و گواهی می‌کند؛ مراکز سطح ۲ اصـطلاحاً RA[2] یـا «مـراکـز مـجاز منطقه‌ای» نامیده می‌شوند زیرا یک ناحیهٔ جغرافیایی مشخص مثل کشور یا قاره را پـوشش مـی‌دهند. البـته ایـن تعاریف، استاندارد و دقیق نیستند و هیچ تعریف دقیقی برای مالکیت هر سطح از این ساختار درختی وجود ندارد. «مراکز مجاز منطقه‌ای» (RA) هویت مراکز صدورگواهینامهٔ دیجیتالی (CAها) را که به منظور صدورگواهینامه‌های X.509 برای سازمانها و اشخاص حقیقی راه‌اندازی شده، گواهی و تائید می‌کند. وقتی «ریشه»، یک RA جدید را تائید و گواهی نمود برای آن مرکز، یک گواهینامهٔ X.509 صادر و در اختیار آن RA قرار می‌دهد که در آن هویت و کلید عمومی آن مرکز، درج و امضاء شده است. به روش مشابه وقتی یک RA یک مرکز CA جدید را تائید می‌کند، این مرکز قادر است برای عموم افراد و سازمانها، گواهینامه‌های دیجیتالی شامل کلید عمومی آنها صادر نماید.



شکل ۸-۲۶. (الف) ساختار سلسله‌مراتبی PKI . (ب) یک زنجیره از گواهینامه‌ها.

---
۱. Certificate Authority ۲. Regional Authority

PKI شبیه به این سناریو عمل می‌کند: فرض کنید آلیس برای مبادلهٔ اطلاعات با باب به کلید عمومی او نیاز دارد، لذا گواهینامهٔ او را پیدا کرده و مشاهده می‌کند که این گواهینامه توسط CA5 [در ساختار شکل ۸-۲۶] صادر و امضاء شده است. آلیس هرگز در مورد CA5 چیزی نشنیده است. او می‌تواند به CA5 مراجعه کرده و از آن بخواهد که هویت قانونی خود را ثابت کند. در پاسخ، CA5 گواهینامهٔ خودش را که توسط RA2 صادر و کلید عمومی CA5 در آن درج شده است، برای آلیس می‌فرستد. حال با کلید عمومی CA5، او می‌تواند گواهینامهٔ باب را [که توسط CA5 صادر شده] بررسی کند و طبعاً تائید می‌شود (در حالی که هنوز هویت RA2 مورد تردید است.). در مرحلهٔ بعدی او به سراغ RA2 رفته و از او می‌خواهد که هویت قانونی خود را اثبات نماید. در پاسخ به این درخواست، یک گواهینامهٔ دیجیتالی دیگر برمی گردد که توسط «ریشه» (عالیترین مرکز تائید گواهینامه‌ها)، امضاء و در آن کلید عمومی RA2 درج شده است. حال آلیس می‌تواند به کلید عمومی باب و هویت او اعتماد کند.

سؤال آخر آنکه آلیس چگونه می‌تواند کلید عمومی «ریشه» را پیدا کند؟ فرض بر آن است که هر کسی کلید عمومی «ریشه» را می‌داند. به عنوان مثال، مرورگر او ممکن است به صورت درونی و در هنگام عرضه، این کلید را در اختیار داشته باشد.

باب جزو افراد صمیمی است! و نمی‌خواهد که آلیس زحمت زیادی بکشد. او می‌داند که آلیس مجبور است گواهینامهٔ CA5 و RA2 را بررسی کند، لذا برای کم کردن زحمت او، این دو گواهینامهٔ مورد نیاز آلیس را بدست آورده و آنها را نیز به همراه گواهینامهٔ خود برای او ارسال می‌نماید. حال آلیس با دانستن کلید **«ریشه»** شروع به بررسی گواهینامهٔ سطح بالای RA2 می‌کند و در صورت صحّت، با استفاده از کلید عمومی موجود در آن، گواهینامهٔ بعدی یعنی CA5 را بررسی می‌نماید. بدین ترتیب آلیس احتیاجی به برقراری ارتباط با هیچ مرکزی برای بررسی صحّت گواهینامه‌ها ندارد. به دلیل اینکه تمام این گواهینامه‌ها امضاء شده هستند براحتی می‌تواند هر گونه تقلب و دستکاری در گواهینامه‌ها را کشف کند. زنجیرهٔ گواهینامه‌های دیجیتالی که نهایتاً به «ریشه» ختم می‌شود گاهاً **«زنجیرهٔ اعتماد»** (chain of trust) یا **«مسیر گواهی»** (Certification Path) نامیده می‌شود. از این تکنیک در عمل به صورت گسترده‌ای استفاده شده است.

البته هنوز یک مشکل باقی است و آن هم در خصوص آن است که چه کسی متولی **«ریشه»** خواهد بود؟ راهکار واقعی آن است که یک «ریشهٔ واحد» نداشته باشیم بلکه تعداد بسیار زیادی «ریشه» (Root) وجود داشته باشد و هر یک برای خود تعدادی RA (مراکز منطقه‌ای) و تعدادی CA (مراکز صدور گواهینامه به افراد) داشته باشند. در حقیقت در مرورگرهای جدید، کلید عمومی بیش از صد «ریشه» گنجانیده شده و به صورت پیش‌فرض مشخص است؛ به این مجموعه از کلیدهای عمومی اصطلاحاً **«لنگرهای اعتماد»** (Trust Anchors) گفته می‌شود و بدین ترتیب از تمرکز بر روی یک مرکز واحد و جهانی پیشگیری خواهد شد.

اما مورد دیگری که بروز می‌کند آن است که چگونه عرضه‌کنندگان مرورگر (شرکتهایی که مرورگرها را طراحی می‌کنند) تصمیم بگیرند که کدامیک از مراکز عالی صدور گواهینامهٔ دیجیتالی (Root) مورد اعتماد و کدام بی‌پشتوانه و مشکوک هستند؟ این مورد به کاربر و سطح آگاهی او بر می‌گردد؛ کاربر خود باید تصمیمی منطقی بگیرد و **«لنگرهای اعتماد»** (Trust Anchors) عرضه شده همراه با مرورگر را چشم بسته تأیید نکند. اکثر مرورگرها به کاربر اجازه می‌دهند تا کلیدهای ریشه را بررسی کند (عموماً در قالب گواهینامه‌هایی که توسط ریشه امضاء شده است) و آنهایی را که به نظرش مشکوک می‌رسد، حذف کند.

فهرستها (Directories)

مورد دیگری که در خصوص ساختار هر PKI وجود دارد آن است که گواهینامه‌ها (و زنجیره‌ای که آنها را به ریشه یا لنگرگاه اعتماد می‌رساند) در کجا ذخیره شوند؟ یک راهکار [illegible]ت که هر کاربر شخصاً گواهینامهٔ خودش را

نگهداری کند. این راهکار مطمئن است (زیرا هیچ راهی برای دیگر کاربران وجود ندارد تا بتوانند گواهینامهٔ امضاء شدهٔ او را بدون آن که کشف شود دستکاری کنند) ولیکن این روش چندان راحت نیست. راهکار دیگری که پیشنهاد شده آن است که از سرویس‌دهندهٔ DNS به عنوان فهرست گواهینامه‌ها استفاده شود زیرا قبل از آن که آلیس بتواند با باب ارتباط برقرار کند، احتمالاً از طریق DNS ، آدرس IP او را جستجو می‌کند؛ پس چرا DNS زنجیرهٔ کامل گواهینامه‌های باب را به همراه آدرس IP او نفرستد؟

برخی فکر می‌کنند این همان راهی است که باید ادامه پیدا کند ولی برخی دیگر ترجیح می‌دهند که یک سرویس دهندهٔ اختصاصی برای مدیریت فهرستها وجود داشته باشد که صرفاً گواهینامه‌های X.509 را مدیریت کند. چنین سرویس دهنده‌ای می‌تواند سرویس جستجوی نام را براساس اسامی X.500 فراهم نماید. به عنوان نمونه چنین سرویس دهنده‌ای از دیدگاه تئوری قادر خواهد بود به پرسشی نظیر این مثال پاسخ بدهد: «فهرستی از افراد که نام آنها آلیس است و در بخش فروش شرکتهایی در آمریکا یا کانادا کار می‌کنند به من بده!» سیستم LDAP می‌تواند نامزد مناسبی برای نگهداری اینگونه اطلاعات باشد.

ابطال گواهینامه (Revocation)

دنیای واقعی پر از گواهینامه‌های مختلف مثل گواهینامهٔ رانندگی و پاسپورت است. گاهی اوقات این گواهینامه‌ها می‌تواند باطل شود؛ مثلاً رانندگی در حالت مستی یا برخی دیگر از تخلفات منجر به لغو گواهینامهٔ رانندگی می‌گردد. همین مسائل در دنیای دیجیتال نیز اتفاق می‌افتد: اعطاکنندهٔ گواهینامه به یک شخص یا سازمان ممکن است به دلیل سوءاستفاده از گواهینامه، تصمیم بگیرد که آنرا باطل کند. همچنین ممکن است به دلیل آن که کلید خصوصی صاحب گواهینامه فاش شده یا از آن بدتر کلید خصوصی یک مرکز گواهی امضاء (CA) لو رفته، گواهینامه‌ها باطل شوند. بنابراین یک PKI باید بتواند در صورت لزوم گواهینامه‌ها را لغو کند.

اولین گام در این راستا آن است که هر CA (مرکز صدور گواهینامه) بطور متناوب فهرستی به نام CRL[1] (فهرست گواهینامه‌های باطل شده) صادر و در آن شمارهٔ سریال گواهینامه‌های باطل شده را مشخص نماید. از آنجایی که هر گواهینامه دارای «زمان اعتبار» مشخصی است لذا در CRL شمارهٔ سریال گواهینامه‌هایی درج می‌شود که هنوز منقضی نشده‌اند. زیرا پس از انقضای زمان اعتبار، گواهینامه به صورت خودکار اعتبار خود را از دست می‌دهد و تفاوتی بین گواهینامه‌های منقضی شده یا باطل شده وجود ندارد؛ در هر دو حالت گواهینامه‌ها بلااستفاده هستند.

متأسفانه، کاربرانی که قصد بررسی گواهینامهٔ کسی را دارند باید ابتدا فهرست CRL را بدست بیاورند و بررسی کنند که آیا گواهینامه باطل شده است یا خیر. اگر گواهینامه در این فهرست بود نباید از آن استفاده کرد. ولیکن حتی اگر گواهینامه‌ای در این فهرست وجود نداشته باشد، باز هم امکان دارد بعد از تدوین این فهرست باطل شده باشد. بنابراین مطمئن‌ترین راه تعیین اعتبار، آنست که از CA سؤال شود. دفعهٔ بعد هم که قرار است از همان گواهینامه استفاده شود، باز هم باید از CA سؤال کرد زیرا ممکن است چند ثانیه قبل باطل شده باشد.

یکی دیگر از پیچیدگیهای عمل «ابطال»، آنست که باید بتوان یک گواهینامهٔ باطل شده را تجدید اعتبار کرد. به عنوان مثال گواهینامه‌ای که به دلیل عدم پرداخت هزینهٔ مصوب، باطل شده، پس از پرداخت باید تجدید اعتبار شود. نیاز به ابطال یا تجدید اعتبار گواهینامه‌ها، یکی از بهترین ویژگیهای گواهینامه را پایمال می‌کند که همانا عدم مراجعهٔ مکرّر به CA بوده است.

فهرست CRL کجا باید ذخیره شود؟ بهترین مکان ذخیرهٔ CRL، همان جایی است که اصل گواهینامه‌ها ذخیره

---

۱. Certificate Revocation List

می‌شود. یک راهکار آن است که CA به صورت فعال و متناوب فهرست CRL را منتشر کرده و گواهینامه‌های باطل شده از فهرستهای تمام کاربران حذف شود. اگر از «**سرویس دایرکتوری**» استفاده نشده باشد، می‌توان CRL را در یک نقطهٔ مناسب از شبکه ذخیره کرد؛ از آنجایی که CRL یک سند امضاء شده است هر گونه دستکاری احتمالی در آن، به سادگی کشف خواهد شد و ذخیرهٔ آن در هر نقطه از شبکه خطر امنیتی ندارد.

اگر گواهینامه‌ها مهلت اعتبار بسیار طولانی داشته باشند، CRL نیز بسیار طولانی خواهد شد. به عنوان مثال اگر کارتهای اعتباری، برای پنج سال معتبر باشند تعداد کارتهای باطل شده، نسبت به زمانی که مهلت اعتبار این کارتها سه ماه است صد چندان خواهد بود. یک روش استاندارد برای تعامل با CRL آن است که این فهرست فقط در موارد خاص به صورت یکجا ارسال شود و در عوض فقط آخرین تغییرات در فهرست اعلام گردد. این کار پهنای باند لازم برای توزیع فهرستهای CRL را کاهش خواهد داد.

۸ـ۶ امنیت ارتباطات

تا اینجا مطالعهٔ خود را در خصوص ابزارها و روشهای معمول امنیت به پایان رسانده‌ایم و بسیاری از پروتکلها و تکنیکها را بررسی کرده‌ایم. مابقی این فصل در خصوص آن است که این تکنیکها چگونه در محیط عمل وارد می‌شوند تا امنیت شبکه را تضمین نمایند؛ همچنین در انتهای فصل در خصوص جنبه‌های اجتماعی، امنیتی، نظریاتی را ارائه خواهیم کرد.

در چهار بخش آتی، نگاهی به امنیت ارتباطات (Communication Security) خواهیم انداخت که در خصوص «تحویل مطمئن و سرّی بیتها از مبداء به مقصد، بدون هیچ تغییر یا دستکاری و همچنین چگونگی جلوگیری از تزریق بیتهای ناخواسته به لینک ارتباطی» بحث می‌کند. این روشها اگرچه تنها موارد امنیتی در سطح شبکه نیستند ولی جزو مهمترین موارد محسوب می‌شوند و بالطبع نقطهٔ شروع خوبی خواهند بود.

۸ـ۶ـ۱ IPsec

IETF از سالها قبل بخوبی دریافته بود که «امنیت در اینترنت» در حال زوال است و برقراری امنیت نیز بسادگی میسر نبود زیرا بر سر نقطه‌ای که باید امنیت در آنجا متمرکز می‌شد مناقشه و جدل وجود داشت. بسیاری از خبرگان امنیت بر این اعتقادند که برای تضمین واقعی امنیت در شبکه، رمزنگاری و بررسی صحت پیامها باید «**انتهابه‌انتها**» (End to End) انجام شود (به عبارت دیگر در سطح لایهٔ کاربرد)؛ بدین سیاق، پروسهٔ مبداء اقدام به رمزنگاری داده‌ها و تمهیدات حفاظتی کرده و سپس آنها را برای پروسهٔ مقصد ارسال می‌کند؛ این پروسه نیز داده‌ها را رمزگشایی کرده و آنها را از لحاظ صحّت بررسی می‌کند. هر گونه دستکاری در داده‌ها مابین این دو پروسه، حتی اگر در سطح سیستم عامل انجام شده باشد، قابل کشف است. اشکال این روش آن است که تمام برنامه‌های کاربردی باید به‌گونه‌ای تغییر داده شوند که خودشان امنیت مورد نظر خود را تأمین و تضمین نمایند. با این دیدگاه، رویکرد بهتر آن است که وظیفهٔ رمزنگاری داده‌ها به لایهٔ انتقال محول شود یا آن که لایهٔ جدیدی بین لایهٔ انتقال و لایهٔ کاربرد این وظیفه را بر عهده بگیرد؛ در این صورت باز هم امنیت، «انتهابه‌انتها» خواهد بود، بدون آنکه برنامه‌های کاربردی نیاز به تغییر داشته باشند.

دیدگاه مخالف این رویکرد، آنست که کاربران درک صحیحی از امنیت ندارند و قادر به استفادهٔ صحیح از آن نیستند و در ضمن هیچکس نمی‌خواهد برنامه‌های موجود خود را تحت هیچ شرایطی تغییر بدهد، لذا این لایهٔ شبکه است که باید احراز هویت و رمزنگاری بسته‌ها را (بدون آنکه کاربر را درگیر کند) بر عهده بگیرد. پس از سالها کشمکش بین این دو دیدگاه، نظریهٔ پشتیبانی از امنیت در سطح لایهٔ شبکه به یک پیروزی نسبی دست یافت و استانداردهای امنیت در لایه شبکه شکل گرفت. چکیدهٔ استدلال آن بود که رمزنگاری در سطح لایهٔ شبکه مانع از

انجام صحیح و مطلوب عملیات کاربران آگاه و مسلط به امنیت نخواهد شد در حالی که به کاربران ناآگاه تا حدی کمک می‌کند.[۱]

نتیجهٔ این منازعه طرحی بود که IPsec نامیده شد و در مستندات RFC 2401، 2402 ، 2406 تشریح گشت. از آنجا که تمام کاربران نمی‌خواهند که از رمزنگاری [بسته‌ها] استفاده کنند (زیرا از لحاظ زمان پردازش هزینهٔ بالایی دارد)، لذا استفاده از آن اختیاری است. البته برای آن که طرح IPsec عمومیت خود را از دست ندهد و در همان بدو کار کثرت پروتکل بوجود نیاید تصمیم برآن شد که در تمام حالات رمزنگاری انجام شود ولیکن در عوض، یک الگوریتم «پوچ» (Null Algorithm) (برای آنهایی که نمی‌خواهند بسته‌ها رمزنگاری شوند) تعریف گردید. این «**الگوریتم پوچ**» به دلیل سادگی، راحتی در پیاده‌سازی و سرعت بسیار بالا در RFC 2410 تشریح و از آن ستایش شده است!!!! شاید سریعترین الگوریتم دنیا باشد!

طراحی کامل IPsec متشکل از یک چارچوب کاری (Framework) برای ارائه خدمات چندگانه ، شامل تعدادی الگوریتم و مولفه است. دلیل ارائه چندین رده خدمات (Services) آن است که شاید همه نخواهند برای استفاده از تمام آنها هزینه بپردازند فلذا این خدمات به صورت انتخابی در اختیار کاربران هستند. خدمات ویژه عبارتند از «ارسال محرمانه بسته‌ها» (Secracy)، «تضمین صحّت» (Intgrity) و حفاظت در مقابل حملاتی که‌براساس آنها یک بخش از داده‌ها به صورت تکراری ارسال می‌شوند[۲] (که در آن اخلالگر سعی می‌کند یک سری از بسته‌های مجاز را بصورت تکراری ارسال نماید بدون آن که از محتوای آنها مطلع باشد). تمام خدمات فوق براساس رمزنگاری با کلید متقارن انجام می‌شود زیرا در سطح لایهٔ شبکه کارآیی و سرعت بسیار بالا، کاملاً حیاتی است. دلیل استفاده از چندین الگوریتم رمزنگاری آن بوده که شاید الگوریتمی که امروزه امن به حساب می‌آید در آینده شکسته شود. وقتی IPsec مستقل از الگوریتم خاص طراحی شده باشد، حتی در صورت شکسته شدن یک الگوریتم در آینده، باز هم قابل استفاده خواهد بود و به حیات خود ادامه می‌دهد. [یعنی اعتبار IPsec به اعتبار یک الگوریتم رمزنگاری خاص گره نخورده است.]

دلیل مولفه‌های چندگانه‌ای که این پروتکل دارد آنست که بتوان فقط بر روی یک اتصال TCP متمرکز شد و از داده‌هایی که بین دو ماشین خاص در شبکه مبادله می‌شود مراقبت کرد یا آنکه از بین کل مسیریابها صرفاً ترافیک بین دو مسیریاب امن ، رمزنگاری شود.

یکی از جنبه‌های نسبتاً عجیب IPsec آن است که اگرچه این پروتکل در لایهٔ IP (لایهٔ شبکه) قرار می‌گیرد ولیکن برخلاف IP، اتصالگرا (Connection Oriented) است. در واقع این مسئله چندان هم عجیب و دور از ذهن نیست زیرا برای ایجاد امنیت باید یک کلید رمز بین ماشینها توافق و ایجاد گردد که در اصل نوعی از «**اتصال**» محسوب می‌شود. (زیرا به هماهنگیهای قبلی بین ماشینها نیاز است.)

البته هزینهٔ برقراری چنین اتصالی بر روی حجم زیادی از بسته‌ها سرشکن می‌شود. یک «**اتصال**» در عُرف IPsec اصطلاحاً **SA** (Security Association) نامیده می‌شود. یک SA، اتصالی «**یکطرفه**» بین دو نقطهٔ پایانی در شبکه است که به آن یک «شناسهٔ امنیت» (Security Identifier) منتسب شده است. اگر نیاز باشد که ترافیک در دو جهت به صورت امن مبادله شود، به دو SA احتیاج است. «شناسه‌های امنیت» درون بسته‌هایی که در شبکه و مبتنی بر یک «اتصال» سیر می‌کنند جاسازی شده و از آن برای جستجوی کلید متناظر و همچنین بدست آوردن اطلاعات مرتبط با بسته‌های امن ورودی استفاده می‌شود.

---

۱. به عبارت بهتر کاربرانی که این سطح از امنیت آنها را راضی نمی‌کند هیچ مانعی برای پیاده کردن استراتژی‌های خود در لایه‌های بالاتر نخواهند داشت و می‌توانند این سطح از امنیت را نادیده بگیرند؛ در حالی که برای کاربران معمولی بسیار مفید است. -م.

۲. Replay Attack

با دیدگاه فنّی، IPsec از دو بخش اصلی شکل گرفته است: در بخش اول دو سرآیند (Header) جدید تعریف شده که می‌تواند برای حمل «شناسهٔ امنیت»، داده‌های لازم برای کنترل صحّت اطلاعات و داده‌های مرتبط با امنیت استفاده شود. بخش دیگر یعنی ISAKMP[۱] با ایجاد و توزیع کلید رمز سر و کار دارد. ما به دو دلیل ISAKMP را بررسی نخواهیم کرد، زیرا: (۱) بشدت پیچیده است. (۲) پروتکل اصلی آن IKE[۲] اشکالات بنیانی دارد و باید عوض شود. (رجوع کنید به Perlman & Kaufman, 2000)

از IPsec می‌توان در دو حالت استفاده کرد؛ در **«حالت انتقال»** (Transport Mode) سرآیند IPsec دقیقاً پس از سرآیند بستهٔ IP قرار می‌گیرد و فیلد پروتکل در بستهٔ IP به گونه‌ای مقداردهی می‌شود که مشخص کند پس از سرآیند بستهٔ IP سرآیند IPsec شروع می‌شود.[۳] سرآیند IPsec، شامل اطلاعات امنیتی نظیر شناسهٔ SA، یک شماره ترتیب جدید و احتمالاً تنظیمات لازم برای بررسی صحت محتوای بسته (Payload) است.

در **«حالت تونل»** (Tunnel Mode)، کل بستهٔ IP شامل سرآیند و محتوای آن در درون یک بستهٔ جدید با سرآیند متفاوت جاسازی می‌شود. «حالت تونل» زمانی مفید است که انتهای تونل به جای ماشین مقصد به یک نقطهٔ خاص [مثل یک مسیریاب] ختم شود. در برخی از حالات، انتهای تونل یک ماشین است که نقش «دروازهٔ امنیت» (Security Gateway) را ایفاء می‌کند (مثلاً دیوار آتش یک شرکت). در این حالت، دیوار آتش در حین عبور بسته‌ها، آنها را جاسازی (کپسوله) کرده و طرف دیگر بسته‌ها را باز می‌کند. وقتی تونل به یک ماشین امن در شبکه منتهی شود، ماشینهایی که در درون شبکهٔ محلی آن شرکت واقعند نیازی به IPsec نخواهند داشت. در این حالت فقط دیوار آتش باIPsec سر و کار دارد.[۴] همچنین زمانی که باید مجموعه‌ای از اتصالات TCP به صورت یکجا جمع شده و به صورت جریانی واحد رمزنگاری شوند، **«حالت تونل»** مفید خواهد بود تا اخلالگر متوجه نشود چند بسته و برای چه کسی ارسال می‌شود. گاهی اوقات دانستن آن که چه مقدار ترافیک به کجا می‌رود، اطلاعات با ارزشی محسوب می‌شود. به عنوان مثال در حین بروز یک بحران نظامی، جریان اطلاعات بین کاخ سفید و پنتاگون سریعاً افت کرده و در عوض حجم ترافیک بین پنتاگون و یکی از مقرهای نظامی در کلرادوی آمریکا به همان اندازه افزایش می‌یابد. حال یک جاسوس اخلالگر می‌تواند از این داده‌ها، اطلاعات با ارزشی بدست بیاورد. مطالعهٔ الگوی جریان بسته‌ها، حتی وقتی رمزنگاری شده هستند، اصطلاحاً **«تحلیل ترافیک»** (Traffic Analysis) نامیده می‌شود. «حالت تونل» می‌تواند تا حدودی برای خنثی کردن این مشکل کارساز باشد. اشکال «حالت تونل» آن است که باید یک سرآیند اضافی به هر بستهٔ IP افزوده شود و بدین ترتیب اندازهٔ بسته‌ها افزایش خواهد یافت. برعکس، در «حالت انتقال» اندازهٔ بسته چندان تغییر نخواهد کرد.

اولین سرآیند جدید، AH[۵] است. این سرآیند، بررسی صحت داده و غیرتکراری بودن بسته‌ها را ممکن می‌سازد ولی داده‌ها سرّی نیستند (به عبارت دیگر رمزنگاری صورت نمی‌گیرد). کاربرد AH در «حالت انتقال»، در شکل ۸-۲۷ به تصویر کشیده شده است. در پروتکل IPv4 این سرآیند (AH)، بین سرآیند اصلی بستهٔ IP و سرآیند بستهٔ TCP قرار می‌گیرد. در پروتکل IPv6 نیز این سرآیند به عنوان **«سرآیند اضافی»** (Extension Header) تلقی می‌شود. [به مشخصات IPv6 مراجعه کنید.] ممکن است به نحوی که در شکل مشخص شده، به دلیل الزام در الگوریتم احراز هویت، لازم باشد که به داده‌ها مقداری دادهٔ زائد (Pad) اضافه شود.

---

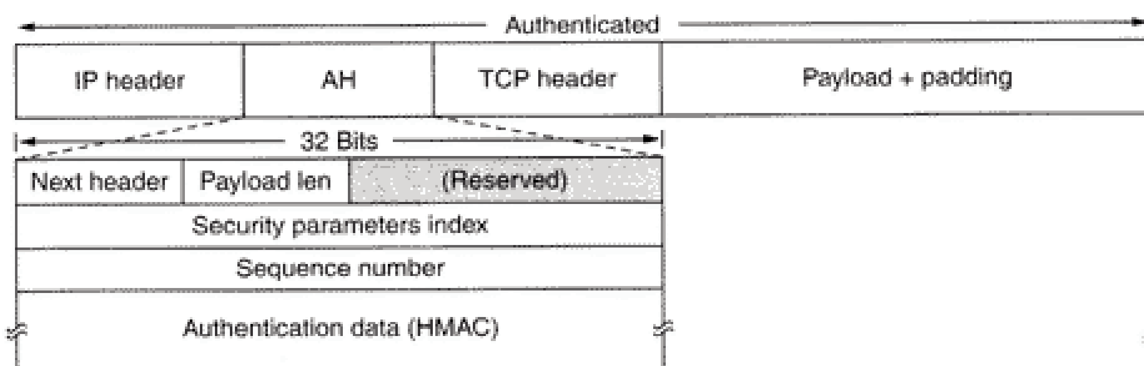
۱. Internet Security Association and Key Management Protocol

۲. Internet Key Exchange

۳. در فیلد پروتکل از بستهٔ IP شمارهٔ پروتکل لایهٔ بالاتر (پردازش‌کنندهٔ بسته) درج می‌شود. درج شمارهٔ IPsec در این فیلد نشان می‌دهد که بستهٔ IP محتوی یک بستهٔ IPsec است، نه بستهٔ TCP، UDP یا هر پروتکل دیگر. رجوع کنید به فصل ۵. -م.

۴. زیرا دیوار آتش به نیابت از همهٔ آنها بسته‌ها را جاسازی و رمزنگاری می‌کند و سمت مقابل، آنها را بازگشایی می‌نماید. بنابراین ماشینها، درگیر باIPsec نخواهند بود.

۵. Authentication Header



شکل ۸-۲۷. «سرآیند احراز هویت» (Authentication Header) که در «حالت انتقال» و برای IPv4 بکار می‌آید.

حال بیایید سرآیند AH را بررسی کنیم: فیلد Next Header بدان منظور به کار می‌رود که مقدار قبلی فیلد Protocol در بستهٔ IP را (قبل از تغییر به مقدار ۵۱[۱]) ، حفظ نماید. در اغلب موارد در این فیلد، عدد ۶ قرار می‌گیرد (به معنای وجود بستهٔ TCP در درون بستهٔ IPsec). فیلد Payload Length، طول سرآیند AH را ۲ واحد کمتر، در مبنای کلمات ۳۲ بیتی نشان می‌دهد.[۲]

فیلد Security Parameter Index ، «شناسهٔ اتصال» است. این فیلد توسط فرستنده تنظیم می‌شود تا رکورد خاصی را در پایگاه اطلاعاتی ماشین گیرنده مشخص کند. این رکورد شامل کلید مشترک و اطلاعات دیگری در خصوص اتصال است. اگر این پروتکل به جای IETF توسط ITU ابداع شده بود این فیلد به نام Virtual Circuit Number نامگذاری می‌شد!

فیلد Sequence Number برای شماره‌گذاری تمام بسته‌هایی است که بر روی یک SA[۳] ارسال می‌شوند. تمام بسته‌ها، حتی آنهایی که به هر دلیل از نو ارسال شده‌اند یک شمارهٔ مستقل و یکتا می‌گیرند. به عبارت دیگر، ارسال مجدد بسته‌ای که قبلاً نیز فرستاده شده با شمارهٔ جدید انجام می‌شود (حتی اگر شمارهٔ ترتیب آن در بستهٔ TCP یکسان باشد.) هدف از این فیلد آن است که «**حملهٔ ارسال تکراری**» (Replay Attack) کشف شود. این شمارهٔ ترتیب هیچگاه از نو به صفر بر نخواهد گشت [اصطلاحاً Wrap around نیست] و هر گاه تمام 2^{32} حالت آن استفاده شد برای ادامهٔ مبادلهٔ اطلاعات باید یک SA جدید به وجود بیاید.

نهایتاً به فیلد Authentication Data می‌رسیم که فیلدی با طول متغیر است و امضای دیجیتالی داده‌های درون بسته در آن قرار می‌گیرد. وقتی یک SA بوجود آمد، ابتدا طرفین در خصوص الگوریتم امضای دیجیتالی که از آن استفاده خواهند کرد مذاکره و توافق می‌کنند. در اینجا عموماً از روشهای مبتنی بر کلید عمومی (Public Key) استفاده نمی‌شود چرا که بسته‌ها باید بی‌نهایت سریع پردازش شوند در حالی که روشهای کلید عمومی بسیار کُند هستند. از آنجایی که IPsec مبتنی بر رمزنگاری با کلید متقارن است و گیرنده و فرستنده قبل از ایجاد SA، بر روی یک کلید مشترک مذاکره و توافق می‌کنند لذا از همین کلید برای محاسبهٔ امضای دیجیتالی استفاده می‌شود. ساده‌ترین راه برای احراز هویت داده‌ها آنست که خلاصهٔ درهم‌شدهٔ (Hash) کلّ بسته به انضمام کلید مشترک استخراج شود و در این فیلد قرار بگیرد. البته کلید مشترک هرگز بر روی خط ارسال نمی‌شود بلکه فقط برای محاسبهٔ این فیلد به بسته اضافه می‌شود. به ساختاری شبیه به این روش، اصطلاحاً HMAC[۴] گفته می‌شود. این

---

۱. مقدار ۵۱ نشانگر وجود بستهٔ IPsec در درون بستهٔ IP است. -م.

۲. طول سرآیند AH متغیر است. -م.

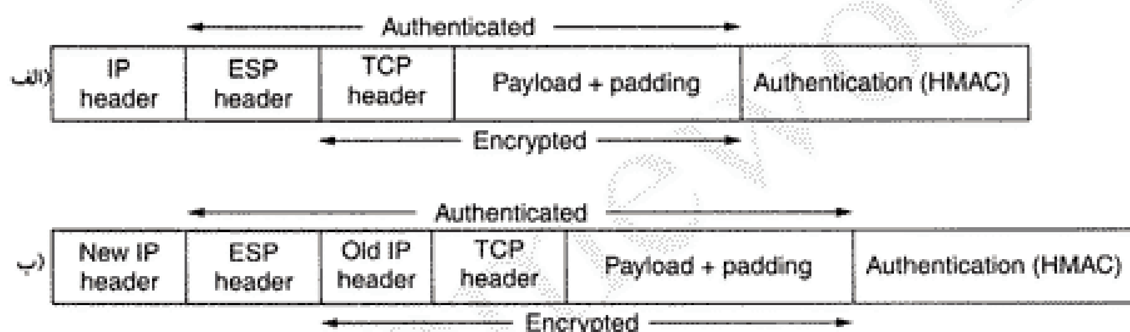
۳. SA را یک نشست یا اتصال یکطرفه فرض کنید. -م.

۴. Hashed Message Authentication Code

روش از نظر حجم محاسبات بسیار سریعتر از آنست که ابتدا SHA-1 اجرا شود و سپس الگوریتم RSA بر روی نتیجهٔ آن اعمال گردد.

سرآیند AH امکان رمزنگاری داده‌ها را فراهم نکرده است و بالطبع زمانی مفید است که فقط صحّت داده‌ها اهمیت داشته باشد و نیازی به ارسال محرمانهٔ داده‌ها نباشد. یکی از ویژگیهای ارزشمند AH آن است که نظارت بر صحت برخی از فیلدهای بسته IP را که بهیچوجه در خلال گذر از یک مسیریاب به مسیریاب دیگر تغییر نخواهند کرد، در برمی‌گیرد. به عنوان مثال فیلد TTL (Time To Live) در بستهٔ IP در هر گام قطعاً تغییر می‌کند فلذا نمی‌توان آن را در محاسبهٔ کُد تایید صحت دخالت داد در حالی که فیلد «آدرس مبداء» (Source Address) می‌تواند در محاسبهٔ کد بررسی صحّت دخالت داده شود تا دستکاری اخلالگران در آدرس مبدأ بسته، نامکن باشد.

یکی دیگر از سرآیندهای بستهٔ IPsec ، «سرآیند ESP» (Encapsulating Security Payload) است. از این سرآیند به نحوی که در شکل ۸–۲۸ نشان داده شده، چه در «حالت انتقال» و چه در «حالت تونل» استفاده می‌شود.



شکل ۸–۲۸. (الف) سرآیند ESP در حالت انتقال. (ب) سرآیند ESP در حالت تونل.

سرآیند ESP، از دو کلمهٔ ۳۲بیتی تشکیل شده است. این دو فیلد عبارتند از: Security Parameter Index و Sequence Number که تعریف آنها را در AH دیدیم. کلمهٔ سومی که عموماً به دنبال این دو فیلد می‌آید (ولیکن جزو سرآیند محسوب نمی‌شود) فیلد Initialization Vector است که برای رمزنگاری اطلاعات کاربرد دارد مگر آن که از الگوریتم رمزنگاری «پوچ» (Null) استفاده شده باشد، در این صورت از آن صرفنظر می‌شود.

ESP برای آزمایش صحت داده‌ها (همانند AH) ، HMAC را عرضه کرده است ولیکن به جای آنکه در سرآیند ظاهر شود، مطابق با شکل ۸–۲۸ پس از فیلد حاوی داده (Payload)، قرار می‌گیرد. قرار دادن HMAC در انتهای بسته، برای پیاده‌سازی سخت‌افزاری مفید خواهد بود زیرا HMAC می‌تواند در خلال ارسال بیتها و خروج آنها از کارت واسط شبکه به صورت سخت‌افزاری محاسبه و در نهایت به انتهای بسته ضمیمه شود. دقیقاً این همان دلیلی است که در شبکه اترنت و دیگر شبکه‌های محلی، کُد CRC به جای آن که در سرآیند فریم ظاهر شود در انتهای فریم می‌آید. با AH (که در ابتدای هر فریم ظاهر می‌شود) بسته ابتدا بافر شده و امضای دیجیتالی آن قبل از ارسال محاسبه می‌شود؛ بدین ترتیب تعداد بسته‌هایی که می‌توان در واحد زمان ارسال کرد کاهش می‌یابد.

هر آنچه که AH می‌تواند انجام بدهد، نه تنها ESP نیز می‌تواند انجام بدهد بلکه کارآیی بیشتر و سرعت بالاتری نیز دارد، پس یک سؤال بوجود می‌آید: چرا با داشتن AH خودمان را به زحمت انداخته‌ایم؟ پاسخ این سؤال، ریشهٔ تاریخی دارد؛ در ابتدا AH فقط صحت داده‌ها (Integrity) و ESP فقط محرمانه ماندن داده‌ها (Secracy) را تضمین می‌کرد. بعداً امکان بررسی صحت داده‌ها به ESP افزوده شد ولیکن گروهی که AH را طراحی کرده بودند نمی‌خواستند که بعد از آن همه کار اجازه بدهند AH فنا شود. تنها دلیل قانع‌کنندهٔ آنها این بود

که AH بخشی از سرآیند بستهٔ IP را در بررسی صحت (Intgrity) بسته دخالت می‌دهد در حالی که ESP این کار را نمی‌کند ولی این دلیل، پشتوانهٔ ضعیفی دارد. یک استدلال ضعیف دیگر آن است که محصولاتی که از AH حمایت می‌کنند ولی از ESP حمایت نمی‌کنند ممکن است مشکلات کمتری در اخذ مجوز صدور از دولت بگیرند زیرا هیچگونه رمزنگاری انجام نمی‌شود. [محصولات مبتنی بر رمزنگاری گاه محدودیتهای دولتی برای صدور دارد.] احتمالاً در آینده AH از صحنه خارج خواهد شد.

۸ـ۶ـ۲ دیوارهای آتش (Firewalls)

این قابلیت که بتوان یک کامپیوتر را در هر کجا به کامپیوتری دیگر در جایی دیگر متصل کرد، به منزلهٔ یک سکهٔ دو رو است؛ برای اشخاصی که در منزل هستند گردش در اینترنت بسیار لذت‌بخش است در حالی که برای مدیران امنیت در شرکتها، یک کابوس وحشتناک به حساب می‌آید. اغلب شرکتها دارای حجم عظیمی از اطلاعات محرمانه و «روی خط» (Online) هستند، مثل اسرار بازرگانی، طرحهای توسعهٔ محصولات، استراتژیهای فروش، تحلیل‌های اقتصادی و نظایر آنها. دسترسی رقبا به این اطلاعات می‌تواند تبعات بسیار سهمگینی داشته باشد.

گذشته از خطر نشت اطلاعات به بیرون و افشای اطلاعات داخلی، خطر نفوذ اطلاعات مخرب به درون نیز وجود دارد. بالاخص ویروسها، کرمها و دیگر آفتهای دیجیتالی، می‌تواند امنیت را درهم بشکند، داده‌های ارزشمند را نابود کند و وقت بسیار زیادی از مسئول شبکه برای ساماندهی به آسیبهای بجا مانده، تلف نماید. این اطلاعات مخرب توسط کارمندان بی‌دقتی که مثلاً خواسته‌اند یک بازی کامپیوتری جدید و جذاب را اجرا کنند، به درون شبکه منتقل می‌شود.

در نتیجه به مکانیزمهایی نیاز است که بتوان بیتهای «خوب» را از بیتهای «بد» تفکیک کرد. یک روش آن است که از IPsec استفاده شود. IPsec از داده‌هایی که بین سایتها در حال تردد هستند بخوبی مراقبت می‌کند ولیکن هیچ کاری برای پیشگیری از ورود آفتهای دیجیتالی (مثل ویروسها) و اخلالگران به درون شبکهٔ محلی شرکت انجام نمی‌دهد.

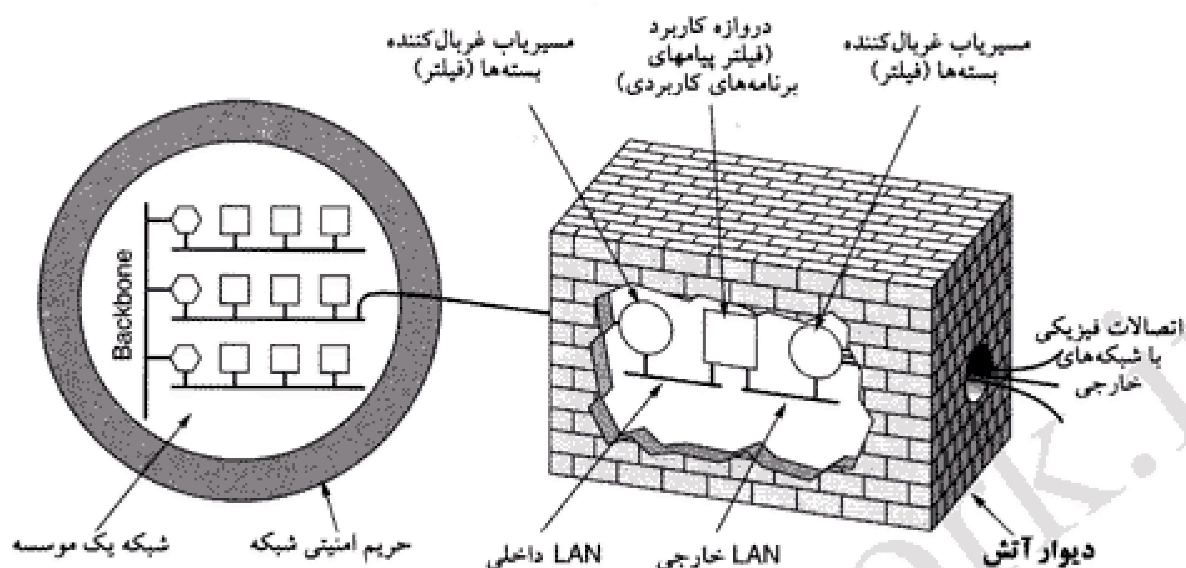
«دیوار آتش» (Firewall) پیاده‌سازی مدرنی از روش قدیمی و قرون وسطایی حصارهای امنیتی است: خندقی عمیق دور تا دور قلعهٔ خود حفر کنید! این الگو همه را مجبور می‌کند تا برای ورود یا خروج از قلعه، از یک پل متحرک و واحد بگذرند و بتوان همه را توسط پلیس حراست بازرسی کرد. در دنیای شبکه‌های کامپیوتری، همین راهکار ممکن خواهد بود: یک شرکت می‌تواند هر تعداد شبکهٔ محلی داشته باشد که به صورت دلخواه به هم متصل شده‌اند، اما تمام ترافیک ورودی یا خروجی شرکت صرفاً از طریق یک پل متحرک (همان دیوار آتش) میسر است (شکل ۸–۲۹ را ببینید.).

دیوار آتش با پیکربندی شکل ۸–۲۹ دو مولفه دارد: (۱) یک جفت مسیریاب که عمل غربال‌سازی بسته‌ها را انجام می‌دهند. (Packet Filtering) (۲) دروازهٔ برنامه‌های کاربردی (Application Gateway). ساختار ساده‌تری نیز وجود دارد ولیکن حُسن بزرگ این طرح آنست که هر بسته باید از دو مرحلهٔ غربال‌سازی (فیلترینگ) و یک مرحله بازرسی محتوایی توسط دروازه، بگذرد. هیچ مسیر دیگری نیز وجود ندارد. خوانندگانی که فکر می‌کنند فقط یک مرحله بازرسی امنیتی کافی است، به احتمال زیاد اخیراً یک پرواز بین‌المللی با خطوط هوایی نداشته‌اند!

هر غربال کنندهٔ بسته، یک مسیریاب استاندارد با برخی از ویژگیهای بیشتر (درخصوص غربال‌سازی)[1] است. این قابلیت اجازه می‌دهد که تمام بسته‌های ورودی و خروجی بازرسی شوند. بسته‌هایی که بتوانند برخی از معیارها

---

۱. تمام مسیریابهای امروزی قابلیت فیلترینگ را دارند. –م.



شکل ۸-۲۹. یک دیوار آتش شامل دو «فیلترکنندهٔ بسته» و یک «دروازهٔ کاربردی».

و شرایط را احراز کنند بطور طبیعی هدایت می‌شوند و آنهایی که در این بازرسی مردود شوند حذف می‌گردند.

در شکل ۸-۲۹، غربال‌کنندهٔ داخلی (متصل به LAN) بسته‌های خروجی از شبکه را و غربال کنندهٔ بیرونی [متصل به خط ارتباط بیرونی] بسته‌های ورودی به شبکه را بازرسی می‌کنند. بسته‌هایی که بتوانند از اولین مانع عبور کنند، برای بازرسی بیشتر واردِ **«دروازهٔ برنامه‌های کاربردی»** می‌شوند. قرار دادن دو غربال‌کننده این تضمین را می‌دهد که هیچ بسته‌ای نتواند قبل از بررسی‌های ابتدایی و سپس گذر از دروازه، از شبکه خارج یا به آن وارد شود.

غربال کنندهٔ بسته عموماً بنابر جدولی که توسط مسئول شبکه تنظیم می‌شود، در خصوص بسته‌ها تصمیم‌گیری می‌کند. در این جدول آدرس مبداء یا آدرس مقصد ماشینهای مجاز و ماشینهای غیرمجاز و همچنین قواعدی در خصوص شرایط تردّد بسته‌ها درج می‌شود.

در شبکه‌هایی که عموماً با TCP/IP پیکربندی شده‌اند مبداء و مقصد با آدرس IP و شماره پورت مشخص می‌شود. شمارهٔ پورت مشخص می‌کند که چه سرویسی مورد نظر است. به عنوان مثال پورت شمارهٔ ۲۳ از TCP متعلق به سرویس TelNet، پورت ۷۹ از TCP متعلق به سرویس Finger و پورت ۱۱۹ از TCP متعلق به سرویس خبررسانی یوزنت می‌باشد. یک شرکت می‌تواند تمام بسته‌ها با هر آدرس IP را که با یکی از شماره پورتهای بالا ترکیب شده است، حذف نماید. در این حالت هیچ شخصی در خارج از شرکت قادر نخواهد بود که از طریق TelNet به یک ماشین وارد شود (Log کند) یا از طریق سرویس Finger فهرست افرادی را که در حال کار با شبکه هستند، بدست بیاورد. همچنین یک شرکت باید با حذف بسته‌های **یوزنت**، مانع از آن شود که کارمندانش روز خود را با خواندن اخبار بگذرانند.

مسدود و حذف کردن بسته‌های خروجی از شبکه نیاز به زیرکی بیشتری دارد زیرا اگرچه اکثر سایتها خودشان را مقیّد به شماره‌گذاری استاندارد پورتها کرده‌اند ولی اجباری به انجام این کار نیست.[1] گذشته از آن، در سرویسهای بسیار مهمی نظیر FTP (پروتکل انتقال فایل) شماره پورت به صورت پویا تعیین می‌شود. اگرچه

---

۱. مثلاً اغلب سایتها از پورت ۸۰ صرفاً برای سرویس‌دهندهٔ وب استفاده می‌نمایند ولی برخی از افراد برای ردّ گم کردن شمارهٔ پورت ۸۰ را برای سرویس‌دهنده‌های دیگر (مثل Telnet یا NetCat) انتخاب کرده‌اند. -م.

مسدود ساختن اتصالات TCP (از طریق حذف بسته‌ها) کاری مشکل است، حذف بسته‌های UDP حتی از آن هم دشوارتر است چراکه هیچ آگاهی اولیه در خصوص آن که بسته چه کاری انجام خواهد داد وجود ندارد. بسیاری از غربال کننده‌های بسته به گونه‌ای پیکربندی شده‌اند که به سادگی ورود و خروج بسته‌های UDP را در دو جهت مسدود و قدغن کنند.

نیمهٔ دوم یک دیوار آتش «**دروازهٔ برنامه‌های کاربردی یا** Application Gateway» است. این قسمت به جای آن که بررسی خود را بر روی مشخصات بسته‌های خام متمرکز کند در سطح لایهٔ کاربرد عمل می‌کند. مثلاً دروازهٔ پست الکترونیکی (Mail Gateway)، دروازه‌ای است که براساس مشخصات هر پیام تصمیم می‌گیرد که آیا ورود یا خروج آن مجاز است یا خیر! برای هر پیام، «دروازه» با بررسی فیلدهای سرآیند پیام، طول پیام یا حتی محتوای پیام، تصمیم به حذف یا ارسال آن می‌گیرد. (به عنوان مثال در یک مقر نظامی، وجود کلماتی نظیر «بمب»، «اتمی» در پیام ممکن است به انجام عملیات ویژه بیانجامد.) مؤسسات آزادند که به هر تعداد «دروازهٔ برنامهٔ کاربردی» برای سرویس دهنده‌های خود نصب نمایند ولی بطور کلی تردد نامه‌های الکترونیکی و استفاده از وب جهانی در اغلب سازمانها مجاز شمرده می‌شود. بقیهٔ سرویسها معمولاً مسدود هستند. ترکیب رمزنگاری و غربال‌سازی بسته‌ها می‌تواند ساختاری را ایجاد کند که در آن امنیت در سطحی محدود و در ازای از دست رفتن سهولت (سهولت کاربری و پیکربندی شبکه) بدست آید.

حتی اگر دیوار آتش به درستی پیکربندی شده باشد، باز هم انبوهی از مشکلات امنیتی باقی خواهد ماند. به عنوان مثال اگر دیوار آتش به گونه‌ای پیکربندی شده باشد که فقط به بسته‌های یک شبکه خاص (مثل شبکهٔ متعلق به یکی از کارخانه‌های شرکت) اجازهٔ تردّد بدهد، یک اخلالگر قادر خواهد بود با قرار دادن آدرسهای غلط (آدرس مبداء جعلی) از حصار این بازرسی عبور نماید. اگر یکی از افراد در داخل شبکه بخواهد یک سند محرمانه را خارج نماید می‌تواند آن را رمز کند و یا حتی به یک تصویر تبدیل کرده و آن را در قالب یک فایل JPEG ارسال نماید تا از هر غربال کننده‌ای که حتی درون متن را جستجو می‌کند، بگذرد. هنوز این حقیقت را خاطرنشان نکرده‌ایم که ۷۰ درصد از کل حملات از درون شبکه و قبل از دیوار آتش صورت می‌گیرد که به عنوان مثال توسط کارمندان ناراضی هدایت می‌شود. (Schneier, 2000)

به علاوه، یک ردهٔ کامل از حملات وجود دارند که دیوار آتش هیچ کاری در مواجهه با آنها نمی‌تواند انجام بدهد. ایدهٔ اصلی دیوار آتش آن است که جلوی ورود اخلالگران را به شبکه و خروج اطلاعات محرمانه از آن را بگیرد. متأسفانه افرادی هستند که هیچ کاری برایشان بهتر از آن نیست که یک سایت خاص را از کار بیندازند. آنها این کار را با ارسال بسته‌های مجاز در تعداد بسیار زیاد به یک هدف در شبکه انجام می‌دهند تا هدف زیر بار بالا درهم بشکند. به عنوان مثال برای زمین‌گیر کردن یک سایت وب، اخلالگر می‌تواند حجم بسیار زیادی بسته‌های TCP SYN را برای برقراری اتصال TCP به سوی آن ماشین روانه کند. سایت مربوط جدولی را (به ازای هر تقاضای اتصال) تخصیص داده و در پاسخ بستهٔ SYN+ACK باز پس می‌فرستد. اگر اخلالگر هیچ واکنشی به این بسته‌های پاسخ نشان ندهد، جدول تخصیص داده شده برای چندین ثانیه و تا زمان انقضای مهلت، در حافظه باقی خواهد ماند. اگر اخلالگر بتواند هزاران بستهٔ تقاضای اتصال (TCP SYN) را ارسال کند، جدول مربوطه پر شده و از آن به بعد برقراری هیچ ارتباط مُجاز نیز ممکن نخواهد بود. حملاتی که در آنها مقصود اخلالگر به جای سرقت اطلاعات از کار انداختن یک هدف در شبکه است اصطلاحاً «**حملهٔ DoS**»[1] نامیده می‌شود. معمولاً بسته‌های تقاضا دارای آدرس مبداء غلط هستند و بدین ترتیب براحتی نمی‌توان اخلالگر را تعقیب کرد.

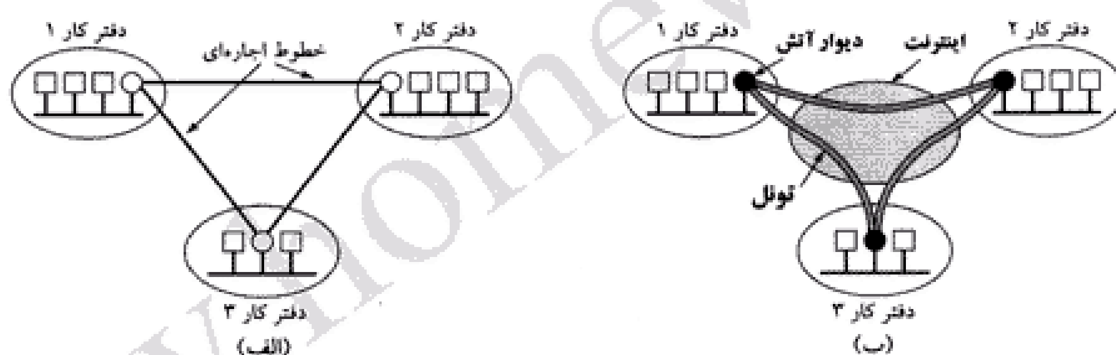
---

۱. حملهٔ اخلال در سرویس‌دهی یا Denial of Service

یک حالت بسیار خطرناکتر آن است که اخلالگر توانسته باشد به صدها کامپیوتر در هر کجای دنیا نفوذ کند و آنها را برای حمله به یک هدف مشترک در یک زمان مشخص، تحت فرمان خود در آورد. این روش نه تنها قدرت حملهٔ اخلالگر را افزایش می‌دهد بلکه احتمال تعقیب و یافتن او نیز کاهش می‌یابد زیرا بسته‌هایی به هدف گسیل می‌شوند که متعلق به ماشین کاربران عادی و غیرمشکوک هستند. به چنین حمله‌ای اصطلاحاً **«حملهٔ DDoS»**[1] گفته می‌شود؛ مقابله با چنین حمله‌ای واقعاً مشکل است. حتی اگر ماشین تحت حمله بتواند سریعاً درخواستهای غیرمتعارف را از درخواستهای مجاز تشخیص بدهد زمانی طول می‌کشد تا پردازش و حذف شوند و اگر این درخواستها در ثانیه، از حدّی بیشتر شود کلّ زمان CPU صرف پردازش آنها خواهد شد.

۸-۶-۳ شبکه‌های خصوصی مجازی (VPN)

بسیاری از شرکتها داری دفاتر و کارخانه‌هایی هستند که در شهرها و گاهی در چندین کشور پراکنده‌اند. در ایام گذشته، قبل از ایجاد شبکه‌های عمومی داده، برای اتصال شبکه‌های پراکندهٔ متعلق به یک شرکت، رایج‌ترین کار استفاده از خطوط اجاره‌ای (Leased Line) متعلق به شرکتهای مخابرات بود. شبکه‌ای که از کامپیوترهای یک شرکت و خطوط اجاره‌ای تلفن تشکیل شده اصطلاحاً **«شبکهٔ خصوصی»** (Private Network) نامیده می‌شود. مثالی از یک شبکهٔ خصوصی که سه شبکه را به هم متصل ساخته در شکل ۸-۳۰-الف نشان داده شده است.



شکل ۸-۳۰. (الف) یک شبکهٔ خصوصی با خطوط اجاره‌ای. (ب) شبکهٔ خصوصی مجازی.

شبکه‌های خصوصی، بسیار خوب و مطمئن عمل می‌کنند. اگر خطوطِ در اختیار شرکت، تماماً اجاره‌ای باشند، هیچ ترافیکی نمی‌تواند به بیرون از شرکت نشت کند و اخلالگر مجبور است به صورت فیزیکی از خطوط انتقال انشعاب گرفته و بدانها متصل شود که انجام این کار نیز ساده نخواهد بود. مشکل بزرگ شبکه‌های خصوصی آنست که اجاره کردن یک خط T1 [با نرخ ارسال 1.544Mbps] در هر ماه هزاران دلار هزینه دارد؛ خطوط T3 نیز چندین برابر گرانتر هستند. وقتی شبکه‌های عمومی داده[2] و بعد از آن اینترنت به صحنه آمد، بسیاری از شرکتها تصمیم گرفتند که انتقال داده‌های خود (و احتمالاً انتقال صوت) را از طریق شبکه‌های عمومی موجود انجام بدهند که هزینهٔ ناچیزی دارد ولیکن در عوض امنیت یک شبکهٔ خصوصی را ندارد.

احساس این نیاز به ابداع VPN (**شبکهٔ خصوصی مجازی**) انجامید که بر روی زیرساخت شبکهٔ عمومی بنا شده است ولی اکثرِ ویژگیهای یک شبکهٔ خصوصی را عرضه می‌کند. این گونه شبکه‌ها از آن جهت **«مجازی»** نامیده شده‌اند که صرفاً یک توّهم (یا بهتر بگوییم یک مدل انتزاعی) هستند؛ دقیقاً مثل **«مدار مجازی»** که در آن هیچ مدار واقعی در کار نیست یا «حافظهٔ مجازی» که در آن هیچ حافظهٔ واقعی از نوع RAM وجود ندارد.

۱. Distributed Denial of Service   ۲. Public Data Network

اگرچه VPN را می‌توان بر روی ATM (یا شبکه Frame Relay) پیاده کرد ولیکن عمومیترین روش آن است که VPN مستقیماً بر روی اینترنت بنا نهاده شود. رایجترین طرح آن است که هر دفتر کار [از یک شرکت] به یک دیوار آتش مجهز شود و به گونه‌ای که در شکل ۸-۳۰-ب نشان داده شده یک تونل بین هر دو دفتر از طریق خطوط عمومی اینترنت ایجاد شود. اگر از IPsec برای ایجاد تونل بین این دو دفتر استفاده شده باشد می‌توان ترافیک جاری بین دو طرف ارتباط را از طریق یک SA[1] که احراز هویت و رمزنگاری شده است، ارسال کرد و بدین ترتیب صحت و امنیت داده‌ها تضمین می‌شود و همچنین ایمنی قابل توجهی در مقابل خطر «تحلیل ترافیک» (که در بخش IPsec بدان اشاره شد) بدست می‌آید.

وقتی این سیستم فعال گردد، زوج دیوار آتش مستقر در دو شبکه باید ابتدا در خصوص پارامترهای SA شامل نوع حالت [حالت انتقال/حالت تونل]، نوع سرویسها و نوع الگوریتم و کلیدها مذاکره و توافق کنند. بسیاری از دیوارهای آتش قابلیت ایجاد VPN را به صورت درونی در خود دارند اگرچه امروزه برخی از مدلهای معمولی مسیریابها نیز همین کار را بخوبی انجام می‌دهند، ولیکن از آنجایی که دیوارهای آتش ذاتاً در محیطهای امن بکار گرفته می‌شوند بنابراین طبیعی است که تونلهایی داشته باشیم که از یک دیوار آتش شروع و به دیگری ختم می‌گردند تا تفکیک کاملی بین اینترنت و شبکهٔ یک شرکت ایجاد نماییم. بدین ترتیب دیوارهای آتش ، VPN و IPsec به همراه ESP (در حالت تونل) ترکیب طبیعی برای چنین محیطهایی است و در عمل به صورت گسترده‌ای از این ترکیب استفاده می‌شود.

پس از آن که SA ایجاد شد، ترافیک می‌تواند جریان پیدا کند. از دیدگاه مسیریابهایی که در درون ساختار شبکهٔ اینترنت واقع هستند بسته‌هایی که از تونل VPN منشاء گرفته‌اند هیچ تفاوتی با بسته‌های معمولی IP ندارند.[2] تنها چیزی که بسته‌های IP حاوی داده‌های معمولی را از بسته‌های IP حاوی بستهٔ IPsec جدا می‌کند سرآیند بستهٔ IPsec است که دقیقاً بعد از سرآیند بستهٔ IP معمولی قرار گرفته است ولی برای مسیریابها این سرآیند اضافی هیچ اهمیتی ندارد و در شرایط عادی تأثیری بر عمل هدایت و مسیریابی نمی‌گذارد چراکه مسیریابها عموماً به این سرآیند اضافی اعتنایی نمی‌کنند.

بزرگترین حسن ایجاد شبکهٔ VPN آن است که بطور کلی از تمام نرم‌افزارهای کاربر مستقل بوده و هیچ تغییری در آنها نیاز نیست و امکانات آن کاملاً نامرئی است: دیوارهای آتش به صورت مستقل SAهای لازم را ایجاد و مدیریت می‌کنند. تنها کسی که از تنظیمات آن اطلاع دارد مسئول شبکه است که طبعاً موظف به پیکربندی و مدیریت دیوارهای آتش می‌باشد. برای بقیهٔ افراد، این شبکه دقیقاً مشابه با شبکه‌های خصوصی با خطوط اجاره‌ای است. برای کسب اطلاعات بیشتر در خصوص VPN مرجع (Brown, 1999; Izzo, 2000) را نگاه کنید.

۸-۶-۴ امنیت شبکه‌های بی‌سیم

از لحاظ منطقی طراحی سیستمی که کاملاً امن باشد بکمک VPN و دیوار آتش، کاری بسیار ساده است ولیکن در عمل شبیه به یک آبکش، اطلاعات از آن نشت خواهد کرد! این وضعیت زمانی اتفاق می‌افتد که برخی از ماشینهای شبکه بی‌سیم بوده و از مخابرهٔ رادیویی استفاده کرده باشند که در این صورت در همهٔ جهات پیرامون دیوار آتش، داده‌های در حال تبادل قابل شنود هستند. محدودهٔ کاری شبکهٔ ۸۰۲.۱۱ حدود چند صد متر است، لذا هر کسی که بخواهد جاسوسی یک شرکت را بنماید می‌تواند براحتی خودروی خود را صبح زود در پارکینگ کارمندان پارک کرده و یک کامپیوتر کیفی مجهز به شبکهٔ ۸۰۲.۱۱ را در درون خودرو به گونه‌ای پیکربندی نماید که هر آنچه را می‌شنود ذخیره کند. عصر همان روز، دیسک سخت این کامپیوتر سرشار از اطلاعات با ارزش خواهد بود. البته از

---

۱. Security Association　　۲. در حالت تونل بسته‌های IPsec در درون یک بستهٔ IP معمولی جاسازی می‌شوند. -م.

دیدگاه تئوری فرض بر آن است که هیچ فردی به بانک دستبرد نمی‌زند!!

بسیاری از مشکلات امنیتی در شبکه‌های بی‌سیم به سازندگان ایستگاههای ثابت[1] بر می‌گردد که سعی می‌کنند محصولاتشان هر چه بیشتر ساده و با محیطی دوستانه باشد. عموماً اگر یک کاربر دستگاه جانبی مورد نیاز برای اتصال به شبکهٔ بی‌سیم را خریداری و آن را به برق متصل کند، بلافاصله عملیاتی شده و شروع به کار می‌کند در حالی که هیچگونه امنیتی وجود ندارد و تمام اطلاعات محرمانه در محدودهٔ برد شبکهٔ بی‌سیم فاش خواهد شد. شبکه بی‌سیم رویای جاسوسان الکترونیکی را محقق کرده است یعنی: «دسترسی آزاد به داده‌ها بدون نیاز به انجام هیچ کاری!»

بنابراین باید اشاره کرد که امنیت اطلاعات در شبکه‌های بی‌سیم از اهمیت و حساسیت بیشتری نسبت به شبکه‌های مبتنی بر سیم برخوردار است. در این بخش به روشهایی که سعی در برقراری امنیت در شبکه‌های بی‌سیم دارند نگاهی خواهیم انداخت. اطلاعات بیشتر را می‌توانید در (Nichols and Lekkas, 2002) بیابید.

امنیت شبکه ۸۰۲.۱۱

استاندارد ۸۰۲.۱۱ پروتکلی برای ایجاد امنیت در سطح لایهٔ پیوند داده‌ها به نام WEP[2] معرفی کرده که هدف از طراحی آن، برقراری امنیت در شبکه‌های بی‌سیم در سطحی معادل با شبکه‌های مبتنی بر سیم بوده است. از آنجایی که شبکه‌های محلی مبتنی بر سیم، به صورت پیش فرض هیچ امنیتی ندارند رسیدن به این هدف بسیار ساده است و قطعاً WEP (به گونه‌ای که خواهیم دید) به این هدف رسیده است!!!!!

وقتی گزینهٔ امنیت (WEP) در شبکهٔ بی‌سیم ۸۰۲.۱۱ فعال شده باشد، هر ایستگاه دارای کلیدی مشترک با ایستگاه ثابت (Base Station) خواهد بود. چگونگی توزیع این کلیدها در استاندارد WEP تعریف نشده است. ممکن است این کلیدها توسط سازندهٔ سخت‌افزار بی‌سیم به صورت کاملاً تصادفی انتخاب و از قبل تنظیم شده باشد. بعداً می‌توان این کلیدها را تعویض کرد. نهایتاً چه ایستگاه ثابت و چه ماشین کاربران (ایستگاههای متحرک) می‌توانند از طریق این کلیدِ از قبل تعیین شده، یک کلید تصادفی برای خود انتخاب کرده و پس از رمزنگاری، آنرا برای یکدیگر بفرستند. پس از ایجاد و توافق، این کلید می‌تواند برای ماهها یا حتی سالها ثابت باقی بماند.

WEP برای رمزنگاری از روش Stream Cipher[3] و الگوریتم RC4 استفاده می‌کند. RC4 توسط رونالد ری‌وست (از ابداع کنندگان RSA) طراحی شده و الگوریتم آن محرمانه نگاه داشته شده بود تا آن که در سال ۱۹۹۴ این الگوریتم کشف و در اینترنت اعلام شد! قبلاً هم اشاره کرده بودیم که محرمانه نگه داشتن الگوریتم تقریباً غیرممکن است؛ حتی وقتی هدف آن باشد که ویژگیهای عالی آنرا از دیگران مخفی نگاه داریم! (در مورد RC4 نیز هدف همین بود.) الگوریتم RC4 بکار رفته در WEP یک Keystream تولید کرده و آنرا با داده‌های رمز نشده XOR می‌کند تا متن رمزشده بدست آید.

فیلد دادهٔ هر بستهٔ اطلاعاتی طبق روشی که در شکل ۸-۳۱ نشان داده شده رمز می‌شود. ابتدا از اصل پیام با استفاده از یک چند جمله‌ای CRC-32، یک کد چهار بایتی کشف خطا استخراج و به انتهای داده‌ها ضمیمه می‌شود تا مجموع این دو به الگوریتم رمزنگاری تحویل گردد. سپس این داده‌های رمز نشده با یک Keystream طولانی XOR می‌شود. حاصل این XOR، داده‌های رمز شده خواهد بود. IV[4] مورد نیاز برای شروع در الگوریتم RC4 به همراه داده‌ها ارسال خواهد شد. وقتی گیرنده بسته‌ای را دریافت می‌کند داده‌های رمزنگاری شده را از درون آن استخراج نموده و براساس IV ارسالی و همچنین کلید مشترک، Keystream را محاسبه کرده و برای رمزگشایی

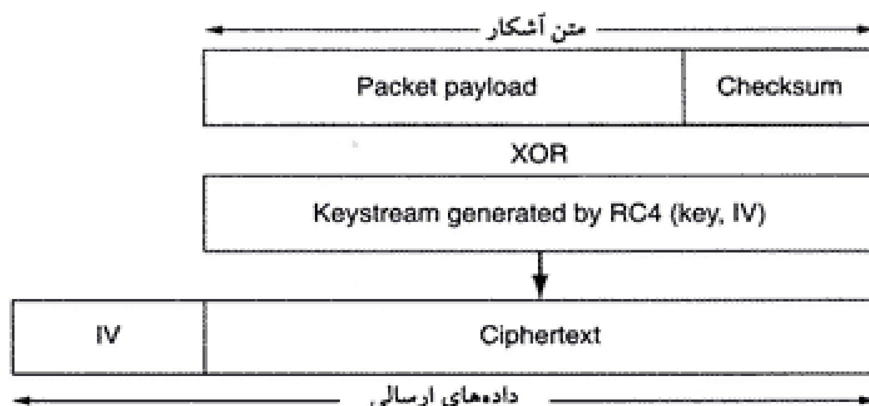
---

۱. ایستگاههای ثابت (Base Stations) نقاط دسترسی ایستگاههای رادیویی به شبکه هستند.

۲. Wired Equivalent Privacy

۳. بخش ۸-۲-۳ را ببینید.

۴. Initialization Vector



شکل ۸-۳۱. رمزنگاری بسته با استفاده از WEP.

اطلاعات، آن را با محتوای بسته XOR می‌کند. سپس برای بررسی هر گونه دستکاری در داده‌ها، کد کشف خطای آن [یعنی CRC] را آزمایش و بررسی می‌نماید.

اگرچه در نگاه اول این روش بسیار خوب به نظر می‌رسد ولی راهی برای شکستن آن تدوین و پیشنهاد شده است. (Borisov et al., 2001) در ادامه خلاصه‌ای از روش درهم شکستن WEP را ارائه می‌کنیم. اول از همه آنکه در بسیاری از مؤسسات، کلیدی مشترک و یکسان برای همهٔ کاربران تعریف شده که در چنین حالتی یک کاربر می‌تواند ترافیک تمام کاربران دیگر را براحتی بدست آورده و بخواند. این مسئله دقیقاً مشابه با شبکهٔ اترنت است ولی به هیچوجه امن نیست.

حتی اگر هر کاربر کلیدی مجزا داشته باشد باز هم می‌توان به WEP حمله کرد. از آنجایی که کلیدها برای مدت زمانی بسیار طولانی تغییر نمی‌کنند، WEP توصیه کرده که لااقل IV برای هر بسته تغییر کند تا نتوان از طریق حملهٔ Keystream Reuse Attack که شرح آن در بخش ۸ـ۲ـ۳ آمد، اطلاعات را رمزگشایی کرد. (تغییر IV فقط توصیه شده ولی اجباری نیست.) متأسفانه در بسیاری از کارتهای واسط شبکهٔ ۸۰۲.۱۱ که برای کامپیوترهای کیفی ارائه شده، وقتی کارت در درون کامپیوتر قرار داده می‌شود، مقدار IV به صفر تنظیم و به ازای ارسال هر بسته یک واحد به IV اضافه می‌شود. از آنجایی که افراد، این کارت را باز و بسته می‌کنند، لذا مقدار IV عموماً کم است. اگر ترودی بتواند تعدادی بسته را که با IV مشابه توسط یک کاربر خاص ارسال شده، جمع‌آوری کند براحتی قادر خواهد بود محتوای دو بسته را با هم XOR کرده و با حذف کلید احتمالاً رمز آنها را بشکند. (فراموش نکنید که IV به صورت آشکار و به همراه بسته ارسال می‌شود.)

ولیکن حتی اگر کارت شبکه ۸۰۲.۱۱ برای هر بسته یک IV مستقل و تصادفی انتخاب کند باز هم ممکن است از IV مشابه استفاده شود چراکه IV جمعاً ۲۴ بیت است و پس از ارسال $۲^{۲۴}$ بسته، از شماره‌های تکراری استفاده خواهد شد. بدتر از آن، اگر IV به صورت تصادفی انتخاب شود طبق روشی که در بخش ۸-۴-۴ در مورد **«حملهٔ روز تولد»** اشاره شد، متوسط بسته‌هایی که باید ارسال شود تا به یک زوج IV مشابه برخورد کنیم حدود ۵۰۰۰ است. ($۲^{۲۴/۲}=۲^{۱۲}=۴۰۹۶$) طبق این استدلال، اگر ترودی برای چند دقیقه به بسته‌های در حال مبادله گوش بدهد قادر خواهد بود حداقل دو بسته با IV یکسان و کلید مشابه بدست آورد. با XOR کردن دو بسته کلید حذف شده و حاصل XOR اصل دو پیام، بدست می‌آید. این رشتهٔ بیت را می‌توان به روشهای مختلف مورد حمله قرار داد تا اصل پیامها بازیابی شود. با اندکی کار بیشتر، Keystream متناظر با آن IV بدست خواهد آمد. ترودی می‌تواند کار خود را اندکی ادامه بدهد و یک دیکشنری برای Keystream متناظر با تمام IVها تشکیل دهد. پس از شکسته شدن IV، تمام بسته‌هایی که در آینده ارسال خواهد شد (یا حتی در گذشته ارسال شده) براحتی قابل رمزگشایی

است. گذشته از آن، چون که IVها به صورت تصادفی بکار می‌روند به محض آن که ترودی بتواند یک زوج (Keystream و IV) را تعیین کند، قادر خواهد بود بسته‌های دلخواه خود را با آن رمز کرده و به صورت جعلی برای یکی از طرفین بفرستد و بدین ترتیب در مبادلهٔ اطلاعات بین کاربر و ایستگاه ثابت مداخله نماید. از دیدگاه تئوری، برای کشف این موضوع، گیرنده باید تمام بسته‌هایی که به ناگاه و با IV مشابه ارسال می‌شوند را بررسی نماید تا از این حمله آگاه گردد ولیکن دو اشکال وجود دارد: اول آن که WEP اجازهٔ چنین کاری را داده است[1]؛ دوم آنکه هیچ کس چنین بررسی و آزمایشی را انجام نمی‌دهد!

در آخر باید اشاره کنیم که CRC هیچ کار با ارزشی انجام نمی‌دهد چرا که ترودی قادر است محتوی درون هر بسته را تغییر داده و CRC متناظر با آن را تولید و به آن بیفزاید، بدون آن که لازم باشد برای این کار اطلاعات از رمز خارج شود. کوتاه سخن آن که شکستن ۸۰۲.۱۱ تقریباً ساده و سرراست است. ما به تمام فهرست حملاتی که آقای Borisov کشف کرده نپرداختیم.

در آگوست سال ۲۰۰۱، یک ماه پس از انتشار مقالهٔ Borisov، یک حملهٔ ویرانگر بر علیه WEP توسط Fluhrer، تدوین و گزارش شد. این شخص کشف کرد که بسیاری از کلیدهای بکار رفته برای رمزنگاری دارای ویژگی خاصی هستند که می‌توان از روی Keystream، برخی از بیتهای کلید را استخراج کرد. اگر این حمله چندین بار تکرار شود، استخراج تمام بیتهای کلید با تلاش بسیار کمی ممکن خواهد بود. البته به غیر از ارائه تئوریک این ادعا، Fluhrer و گروهش هیچ تلاشی برای شکستن یک شبکهٔ محلی مبتنی بر ۸۰۲.۱۱ نکرده بود.

در مقابل، وقتی یک دانشجوی کارآموز و دو محقق در آزمایشگاه AT&T از حملهٔ گزارش شده توسط Fluhrer آگاه شدند تصمیم گرفتند آن را در عمل آزمایش کنند. پس از یک هفته تلاش، آنها توانستند اولین کلید ۱۲۸ بیتی را بر روی یک محصول واقعی ۸۰۲.۱۱ پیدا کنند. چندین هفته نیز به دنبال یافتن کارتهای سخت‌افزاری ۸۰۲.۱۱، تهیه مجوز خرید و نصب و آزمایش آنها بودند. برنامه‌نویسی لازم فقط دو ساعت طول کشید!

وقتی این پژوهشگران نتایج کار خود را اعلام کردند، CNN گزارشی خبری تحت عنوان "*Off-the-shelf Hack Breaks Wireless Encryption*" به چاپ رساند که در آن برخی از صنایع، نتایج کار آنها را به مسخره گرفته بودند؛ با ذکر این نکته که کار آنها براساس تئوری Flahrer بوده و کاملاً بدیهی و بی‌ارزش است. اگرچه استدلال این صنایع از دیدگاه فنی درست است ولی این حقیقت را نمی‌توان نادیده گرفت که تلاش مشترک این دو گروه پرده از یک اشکال اساسی در ۸۰۲.۱۱ برداشت.

در هفتم سپتامبر ۲۰۰۱، IEEE با اقرار به این واقعیت که WEP بطور کامل قابل شکستن است با صدور یک اعلامیه کوتاه شامل شش بند جوابیه‌ای را منتشر ساخت که می‌توان آن را به صورت زیر خلاصه کرد:

۱. ما گفته بودیم که امنیت WEP بهتر از امنیت در شبکهٔ اترنت نیست!!
۲. همین امنیت ناقص بهتر از نبود آنست!
۳. سعی کنید از روشهای امنیتی دیگر بهره بگیرید. (مثلاً امنیت در سطح لایهٔ انتقال)
۴. نسخهٔ بعدی، 802.11i امنیت بالاتری خواهد داشت.
۵. صدور گواهینامه برای محصولات ۸۰۲.۱۱ منوط به استفاده از استاندارد 802.11i خواهد بود.
۶. در تلاش هستیم که روشی برای امن کردن آن تا قبل از معرفی 802.11i ارائه بدهیم.

این قضیه را بدان منظور بررسی کردیم تا خاطرنشان کنیم که رسیدن به امنیت کامل حتی برای خبرگان این فن ساده نیست.

---

۱. یعنی WEP ارسال چند بسته با IVهای مشابه را منع نکرده است و ممکن است برخی از کارتهای شبکه چنین کنند. -م.

امنیت در تکنولوژی بلوتوث (Bluetooth)

بلوتوث بُرد کوتاهتری نسبت به شبکهٔ ۸۰۲.۱۱ دارد لذا در این شبکه نمی‌توان با پارک در پارکینگ یک مؤسسه و استراق سمع داده‌ها به آن حمله کرد ولی کماکان امنیت بلوتوث مورد مهمی بشمار می‌رود. به عنوان مثال فرض کنید که کامپیوتر آلیس مجهز به یک صفحه کلید بی‌سیم مبتنی بر بلوتوث است. اگر در این شبکه امنیت وجود نداشته باشد ، ترودی در دفتر مجاور آلیس ، می‌تواند هر آنچه را آلیس تایپ می‌کند (حتی نامه‌های ارسالی او) را بخواند. او همچنین می‌تواند هر آنچه را که آلیس برای چاپ بر روی «چاپگر بلوتوث» می‌فرستد بدست آورد (مثل نامه‌های دریافتی یا گزارشهای محرمانه). خوشبختانه بلوتوث ، ساختار امنیتی دقیقی دارد و تلاش می‌کند تا فعالیتهای اخلالگران دنیا را خنثی کند. در ادامه ویژگیهای اصلی این ساختار را به اختصار بررسی می‌نماییم.

بلوتوث سه حالت امنیتی متفاوت در محدودهٔ «بدون رمزنگاری» تا «رمزنگاری کامل» و «امکان بررسی صحت داده‌ها» دارد. همانند شبکهٔ ۸۰۲.۱۱ هر گاه گزینهٔ امنیت غیرفعال شده باشد (که به صورت پیش فرض اینگونه است) هیچ امنیتی برای داده‌ها وجود ندارد. بسیاری از کاربران گزینهٔ امنیت را غیرفعال نگه می‌دارند تا وقتی که یک اشکال جدی برایشان اتفاق بیفتد؛ پس از آن امنیت را فعال می‌کنند. در دنیای کشاورزی به این بی‌احتیاطی «بستن در اصطبل پس از فرار اسب» گفته می‌شود!!!

بلوتوث امنیت را در چندین لایه عرضه کرده است: در لایهٔ فیزیکی تعویض مستمر فرکانس (Frequency Hopping) امنیت ناچیزی عرضه می‌کند ولیکن از آنجایی که در ابزارهای مبتنی بر این تکنولوژی قبل از پرش به یک فرکانس خاص باید ترتیب تغییر فرکانس به اطلاع طرفین برسد لذا این تغییر محرمانه نیست و برای اخلالگر قابل شنود است. امنیت واقعی زمانی آغاز می‌شود که یک دستگاه «پیرو» (Slave) که تازه به شبکه وارد شده (مثل صفحه کلید مبتنی بر بلوتوث) از دستگاه «اصلی» Master (مثل کامپیوتر) تقاضای یک کانال می‌کند. فرض شده که این دو دستگاه دارای یک کلید مشترک سرّی هستند که از قبل درون آنها درج شده است. در برخی از حالات این کلید مشترک به صورت سخت‌افزاری توسط کارخانهٔ سازنده بر روی آن ابزار ذخیره می‌شود. (به عنوان مثال همانند شماره‌ای که به صورت پیش‌فرض بر روی گوشی تلفن همراه وجود دارد.) در برخی دیگر از حالات یکی از دستگاهها دارای کلیدی تعبیه شده بر روی سخت‌افزار است در حالی که کاربر (برای فعال کردن دستگاه) مجبور است این کلید را در قالب عددی دهدهی وارد کند. این کلیدهای مشترک اصطلاحاً **«کلیدهای عبور»** (Passkeys) نامیده می‌شوند.

برای ایجاد یک کانال، ماشین اصلی (Master) و ماشین «پیرو» (Slave) هر یک بررسی می‌کنند که آیا دیگری کلید عبور را می‌داند؟ در این صورت، با یکدیگر در خصوص آنکه (۱) آیا اطلاعات کانال رمزنگاری شود؛ (۲) صحت آنها بررسی شود (۳) یا هر دو کار انجام شود؛ مذاکره و توافق می‌کنند. سپس یک کلید ۱۲۸ بیتی که برخی از بیتهای آن عمومی و آشکار است برای **نشست** انتخاب می‌نمایند. این نکته که در بلوتوث اجازه داده شده با معلوم بودن برخی از بیتها کلید رمز ضعیف باشد ، بدان دلیل است که در برخی از کشورها طبق قوانین دولتی، اجازهٔ بکارگیری یا صدور محصولاتی که در آنها کلید رمز طولانی است و دولت قادر نیست رمز آن را بشکند، ممنوع است.

رمزنگاری در بلوتوث به روشی مبتنی بر Stream Cipher[1] که E_0 نام دارد، انجام می‌شود. بررسی صحت اطلاعات نیز به روش +SAFER انجام می‌گیرد. این دو روش براساس روش معمولی رمزنگاری با کلید متقارن هستند. +SAFER برای مسابقهٔ AES (که نهایتاً به پیروزی Rijndael انجامید) ارسال شد ولیکن در همان مراحل مقدماتی از دور مسابقه خارج گردید چراکه از بقیهٔ روشهای پیشنهادی کُندتر بود. طراحی شبکهٔ بلوتوث قبل از

---

۱. بخش ۸-۲-۳ را ببینید.

انتخاب برندهٔ AES، پایان یافته بود وگرنه به احتمال زیاد در آن از روش Rijndael استفاده می‌شد.

روش واقعی رمزنگاری بکار رفته در Stream Cipher در شکل ۸-۱۴ نشان داده شده است که در آن متن اصلی با کلید هر مرحله (Stream Key) XOR شده و متن رمز را بدست می‌دهد. متأسفانه E_0 نیز (شبیه به RC4) می‌تواند اشکالات اساسی داشته باشد (Jokobson & Wetzel, 2001). اگرچه هنوز E_0 شکسته نشده است (تا زمان نوشتن این کتاب) ولیکن شباهتهای آن با رمز A5/1 که اِشکال واضح آن ترافیک تلفنهای همراه GSM را در معرض حمله قرار داده است به این نگرانی دامن می‌زند. گاهی این موضوع افراد را سردرگم و متعجب می‌کند که چرا در بازی همیشگی موش و گربه بین متخصصین رمزنگار و رمزشکن، بیشتر اوقات رمزشکنها برنده هستند!

یکی دیگر از موارد امنیتی آن است که بلوتوث صرفاً «دستگاه» را احراز هویت می‌کند نه کاربر را، فلذا یک دستگاه دزدیده شدهٔ مبتنی بر بلوتوث می‌تواند به سارق اجازهٔ دسترسی به حساب کاربری و دیگر امکانات صاحب آن دستگاه را بدهد. با این حال، بلوتوث امنیت را در لایه‌های بالاتر نیز پیاده‌سازی کرده است؛ بنابراین حتی اگر امنیت در لایهٔ پیوند داده نابود شود، در لایه‌های بالاتر باقی خواهد ماند؛ بالاخص در برخی از برنامه‌های کاربردی این ویژگی مثبت وجود دارد که گاهی برای آن که کاربر بتواند کاری را انجام بدهد از او کد PIN (شمارهٔ شناسایی شخصی) مطالبه می‌کنند.

امنیت در WAP 2.0

در اکثر بخشها، مجمع توسعه دهندهٔ WAP از غیراستاندارد بودن پشتهٔ پروتکلی WAP 1.0 درس گرفته و به همین دلیل WAP 2.0 در تمام لایه‌ها به طرز گسترده‌ای از پروتکلهای استاندارد استفاده می‌کند. از آنجایی که WAP مبتنی بر IP است در لایهٔ شبکه به طور کامل از IPsec حمایت می‌کند. در لایهٔ انتقال نیز، از یک اتصال TCP به کمک TLS حفاظت می‌شود. (TLS استاندارد IETF است که در همین فصل بدان خواهیم پرداخت.) در لایهٔ بالاتر از روش «احراز هویت HTTP» که در RFC 2617 تشریح شده است، بخوبی حمایت می‌شود. وجود یک کتابخانهٔ سیستمی (Library) در سطح لایهٔ کاربرد به منظور رمزنگاری، امکانات کافی جهت کنترل صحت و غیرقابل انکار بودن پیامها را در اختیار برنامه‌نویسان WAP قرار داده است. از آنجایی که WAP 2.0 مبتنی بر استانداردهای شناخته شده است این شانس بزرگ وجود دارد که سرویسهای امنیتی آن بالاخص سرویسهای احراز هویت، بررسی صحت داده‌ها، غیرقابل انکار بودن و محرمانه ماندن پیامها، از امنیت 802.11 و Bluetooth بهتر و مطمئن‌تر باشد.

۸ـ۷ پروتکلهای احراز هویت

«احراز هویت» (Authentication) روشی است که براساس آن یک پروسه بررسی می‌کند که آیا شریک او در یک ارتباط (یعنی پروسهٔ طرف مقابل)، همانی است که باید باشد یا یک نفوذگرست که خود را به جای طرف واقعی جا زده است. بررسی هویت واقعی یک پروسهٔ راه دور در شرایطی که با اخلالگران فعال و بدخواه روبرو هستیم فرآیندی بسیار دشوار است و به پروتکلهای پیچیدهٔ مبتنی بر رمزنگاری نیاز دارد. در این بخش برخی از پروتکلهای بی‌شمار احراز هویت را که در شبکه‌های ناامن مورد استفاده قرار می‌گیرد، مطالعه خواهیم کرد.

گاهی مردم اصطلاح **«احراز هویت»** (Authentication) را با **«صدور مجوز»** (Authorization) اشتباه می‌گیرند. «احراز هویت» با این سؤال سر و کار دارد که آیا شما حقیقتاً در حال محاوره و تبادل اطلاعات با یک پروسهٔ خاص هستید. «صدور مجوز» با این مقوله سر و کار دارد که یک پروسه، مجوز انجام چه کارهایی را دارد. به عنوان مثال، یک پروسهٔ مشتری با یک سرویس دهندهٔ فایل ارتباط برقرار کرده و اعلام می‌کند: «من پروسهٔ «اسکات» هستم و می‌خواهم فایل *cookbook.old* را پاک کنم.» از دیدگاه سرویس‌دهندهٔ فایل پاسخ دو سؤال

بایدمشخص شود:

۱. آیا این پروسه حقیقتاً پروسه «اسکات» است؟ (احراز هویت)

۲. آیا «اسکات» اجازهٔ حذف فایل *cookbook.old* را دارد؟ (صدور مجوز)

پس از آن که پاسخ این دو سؤال، بدون هیچ ابهامی مثبت ارزیابی شد، عمل درخواستی قابل انجام است. سؤال اول حساس‌تر و کلیدی‌تر است. پس از آن که سرویس دهندهٔ فایل متوجه شد که با چه کسی صحبت می‌کند، بررسی مجوزها در حد یک جستجوی ساده درون جدول یا پایگاه اطلاعاتی محلی است. به همین دلیل ما در این بخش صرفاً بر روی موضوع احراز هویت متمرکز خواهیم شد.

یک مدل عمومی که تمام پروتکلهای احراز هویت از آن استفاده می‌کنند بدین نحو است که مثلاً: آلیس کارش را با ارسال پیامی به باب یا یک «KDC **مورد اعتماد**»[1] که صادق و امین همه است، شروع می‌کند. چندین پیام دیگر بین طرفین و در دو جهت مبادله می‌شود. ممکن است در حالی که این پیامها در حال مبادله هستند شخص ثالثی مثل ترودی مشغول استراق سمع، دستکاری در پیام یا تکرار پیامها باشد تا بدین نحو آلیس یا باب را فریب بدهد یا در کار آنها اخلال کند.

علیرغم تمام این کارشکنی‌ها، وقتی عملکرد پروتکل تکمیل شده باشد، آلیس مطمئن خواهد بود که در حال صحبت با باب است و باب هم اطمینان دارد که با آلیس محاوره دارد. به علاوه در اغلب پروتکلها، دو طرف یک «**کلید سرّی نشست**» ایجاد می‌کنند تا در محاورهٔ خود از آن [برای رمزنگاری] استفاده نمایند. در عمل و به دلایل سرعت و کارآیی، تمام ترافیک داده‌ها در حین نشست به روش رمزنگاری با کلید متقارن (عموماً AES یا DES) رمز می‌شوند، اگرچه از روش «رمزنگاری کلید عمومی» در پروتکلهای احراز هویت برای ایجاد «کلید نشست» در سطح گسترده‌ای استفاده می‌شود.

دلیل آنکه برای هر «اتصال» جدید یک کلید تصادفی به عنوان کلید نشست انتخاب می‌شود آنست که حجم ترافیکی که مستقیماً توسط کلید سرّی یا کلید عمومی کاربر رمز می‌گردد حداقل بماند و در نتیجه میزان اطلاعات رمز شده (که در تمام آنها از یک کلید استفاده شده) کاهش یابد. همچنین هر گاه پروسه‌ای دچار اشکال شده و درهم بشکند (crash کند)، ممکن است «تصویر حافظهٔ» آن پروسه در اختیار افراد ناباب قرار بگیرد و آنها بتوانند کلید اصلی کاربر را از درون آن استخراج کنند.[2] در این حالت حتی اگر کلیدی فاش شود کلید نشست خواهد بود که آنهم ثابت نیست. تمام کلیدهای ثابت و دائمی پس از آن که نشست برقرار شد از حافظه پروسه‌ها پاک شده و فقط از کلید نشست استفاده می‌شود.

۸-۷-۱ احراز هویت براساس کلید مشترک و سرّی

در اولین پروتکل احراز هویت، فرض می‌کنیم که آلیس و باب قبلاً در مورد یک کلید سرّی به نام K_{AB} با یکدیگر توافق کرده‌اند. ممکن است این کلید را با استفاده از تلفن یا از طریق یک شخص معتمد به اطلاع یکدیگر رسانده باشند ولی در هر حال فرض بر آن است که این کلید سرّی را از طریق شبکهٔ ناامن، برای یکدیگر ارسال نکرده‌اند. این پروتکل بر اصولی استوار است که تمام پروتکلهای دیگر احراز هویت نیز از آن تبعیت می‌کنند: «یکی از طرفین عددی تصادفی برای دیگری ارسال می‌کند و طرف مقابل تبدیل خاصی را بر روی آن اعمال کرده و نتیجه را بر می‌گرداند». چنین پروتکلهایی اصطلاحاً پروتکلهای «**چالش-پاسخ**» (Challenge-Response) نامیده می‌شوند.

---

۱. KDC: مرکز توزیع کلید یا Key Distribution Center

۲. در سیستمهای عاملی مثل یونیکس هرگاه پروسه‌ای دچار اشکال شده و crash کند، کل فضای حافظهٔ در اختیار آن پروسه بر روی دیسک سخت ذخیره می‌شود تا بتوان برای عملیاتی نظیر Recovery از آن بهره گرفت. -م.

در این پروتکل و دیگر پروتکلهای احراز هویت که در ادامه می‌آیند، از نمادهای زیر استفاده شده است:

A و B مشخصه‌های شناسایی [۱] آلیس و باب هستند.

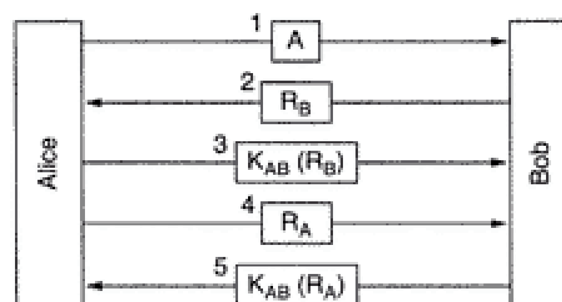
R_i رشته‌های «چالش» (Challenge) هستند که پانویس آنها یعنی i فرستندهٔ آن را مشخص می‌کند.

K_i کلیدهایی هستند که پانویس آنها یعنی i صاحب کلید را مشخص می‌نماید.

K_s کلید نشست

در شکل ۸-۳۲، اولین پروتکل احراز هویت مبتنی بر کلید مشترک و ترتیب مبادلهٔ پیامها نشان داده شده است. در پیام ۱، آلیس مشخصهٔ شناسایی خود یعنی A را به گونه‌ای برای باب می‌فرستد که او بتواند آن را بفهمد (بصورت آشکار و بدون رمزنگاری). البته باب (فعلاً) هیچ راهی برای تشخیص آن که آیا این پیام واقعاً از آلیس آمده یا از شخص ثالثی مثل ترودی، ندارد. به همین دلیل یک عدد تصادفی بسیار بزرگ یعنی R_B را به عنوان رشتهٔ **«چالش»** (Challenge) انتخاب کرده و آنرا در پیام شمارهٔ ۲ بصورت آشکار به آلیس بر می‌گرداند. اعداد تصادفی بکار رفته در پروتکلهای «چالش-پاسخ» مثل این پروتکل، اصطلاحاً *nonce* نامیده می‌شوند. آلیس پیام شمارهٔ ۲ را با کلید مشترک خود رمزنگاری کرده و داده‌های رمز شده یعنی $K_{AB}(R_B)$ را در پیام شمارهٔ ۳ به باب بر می‌گرداند. وقتی باب این پیام را دریافت می‌کند فوراً متوجه می‌شود که این پیام واقعاً از آلیس آمده است زیرا ترودی K_{AB} را نمی‌داند و طبعاً نمی‌توانسته چنین پیامی را تولید نماید. از آنجایی که R_B به صورت کاملاً تصادفی و در یک فضای بسیار بزرگ (مثلاً اعداد تصادفی ۱۲۸ بیتی) انتخاب می‌شود لذا احتمال آن که ترودی قبلاً یکبار R_B و پاسخ آن را مشاهده کرده باشد بسیار بعید است. همچنین تقریباً احتمال آن که او بتواند پاسخ صحیح هر رشتهٔ «چالش» را [بدون داشتن کلید] حدس بزند وجود ندارد.

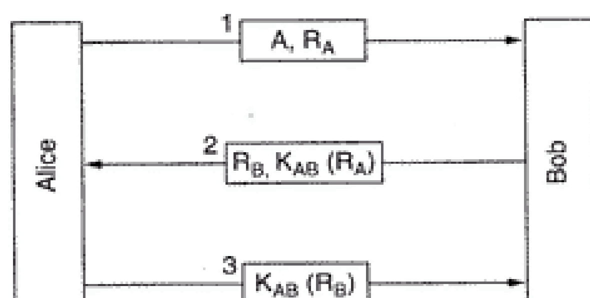
در این جا، باب مطمئن شده که در حال صحبت با آلیس است ولی آلیس از هیچ چیز مطمئن نیست زیرا ممکن است ترودی پیام شماره ۱ را استراق سمع کرده باشد و در پاسخ R_B را برگرداند. اصلاً ممکن است باب، شب قبل فوت کرده باشد!! برای آن که آلیس بداند که در حال صحبت با چه کسی است عدد تصادفی R_A را انتخاب و در پیام شماره ۴ آن را برای باب می‌فرستد. وقتی باب پاسخ $K_{AB}(R_A)$ (یعنی حاصل رمزنگاری R_A با کلید مشترک) را برگرداند، آلیس نیز متوجه می‌شود که واقعاً با باب صحبت می‌کند. حال اگر این دو بخواهند یک کلید نشست ایجاد کنند، آلیس می‌تواند کلیدی مثل K_S را انتخاب و آن را با K_{AB} رمز کرده و برای باب بفرستد.



شکل ۸-۳۲. پروتکل دومرحله‌ای «چالش-پاسخ» جهت احراز هویت.

پروتکل شکل ۸-۳۲ شامل پنج پیام است. حال ببینیم آیا می‌توان با تیزهوشی، تعدادی از این مراحل را حذف کرد. یکی از این راهکارها در شکل ۸-۳۳ نشان داده شده است. در این شکل آلیس به جای آن که منتظر شروع

۱. Identifier

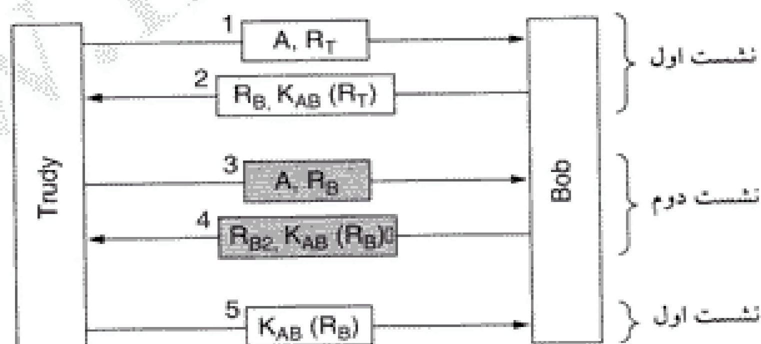


شکل ۸-۳۳. پروتکل دومرحله‌ای احراز هویت با تعداد مراحل کمتر.

مراحل «چالش-پاسخ» توسط باب شود خودش شخصاً این کار را شروع می‌کند.[1] به روش مشابه، باب وقتی به چالش آلیس پاسخ می‌دهد، رشتهٔ چالش خودش یعنی R_B را نیز برای آلیس می‌فرستد. بدین ترتیب کل پروتکل به جای پنج مرحله به سه مرحله کاهش می‌یابد.

آیا پروتکل جدید مزیتی بر پروتکل اصلی دارد؟ به صورت حسّی شاید بگوییم کوتاهتر و سریعتر است. متأسفانه اشتباه است! در شرایط خاص ترودی می‌تواند این پروتکل را با استفاده از تکنیکی نام **«حملهٔ بازتاب»** (Reflection Attack) در هم بشکند. اگر امکان گشودن چند نشست همزمان با باب وجود داشته باشد، ترودی براحتی خواهد توانست این پروتکل را شکست بدهد. این وضعیت زمانی اتفاق می‌افتد که به عنوان مثال، باب یک بانک باشد و طبعاً باید بتواند ارتباط همزمان چندین ماشین خودپرداز (Teller Machine) را بپذیرد.

«حملهٔ بازتاب» که توسط ترودی انجام می‌شود در شکل ۸-۳۴ نشان داده شده است. او حملهٔ خود را با ارسال R_T و اعلام آن که آلیس است ، شروع می‌کند. باب طبق معمول پاسخ این چالش را به همراه رشتهٔ چالش خود یعنی R_B برمی‌گرداند. حال ترودی گیر می‌افتد. او $K_{AB}(R_B)$ را نمی‌داند. پس چه کاری می‌تواند انجام بدهد؟



شکل ۸-۳۴. حملهٔ بازتاب.

او می‌تواند نشست دومی را با ارسال پیام ۳ شروع نماید و R_B بدست آمده از مرحلهٔ دوم را به عنوان رشته **«چالش»** خودش برای باب بفرستد. باب در کمال آرامش آن را رمز کرده و به صورت $K_{AB}(R_B)$ در پیام چهارم برای ترودی می‌فرستد. در شکل ۸-۳۴ ، پیامهای نشست دوم را برای تمایز از نشست اول، خاکستری نشان داده‌ایم. حال ترودی اطلاعاتی را که برای نشست اول کم داشت در اختیار دارد لذا می‌تواند نشست اول را تکمیل کرده و نشست دوم را ناتمام رها کند. حال باب متقاعد شده که ترودی همان آلیس است لذا وقتی حساب بانکی

---

۱. یعنی بلافاصله ضمن معرفی خود، رشتهٔ چالش یعنی R_A را برای باب می‌فرستد.

آلیس را تقاضا می‌کند، باب بی‌هیچ پرسشی اطاعت می‌نماید. وقتی ترودی مثلاً از باب می‌خواهد که تمام موجودی او را به یک حساب محرمانه در سوئیس واریز نماید، باب این کار را بدون اندکی درنگ انجام می‌دهد! پند علمی این داستان آن است که:

«طراحی یک پروتکل صحیح برای احراز هویت دشوارتر از آن است که به نظر می‌رسد.»

چهار قاعدهٔ کلی زیر می‌تواند راهنمای خوبی در طراحی پروتکل باشد:

۱. شروع کننده را وادار کنید که قبل از پاسخ‌دهنده، هویت خود را اثبات کند وگرنه باب قبل از آن که ترودی مدرکی در خصوص هویت خود ارائه داده باشد، اطلاعات با ارزشی را از دست می‌دهد.

۲. شروع کننده و پاسخ دهنده را وادار کنید که از کلیدهای متفاوتی برای اثبات هویت خودشان استفاده کنند حتی اگر این کار به معنای تعریف دو کلید مشترک و مستقل K_{AB} و K'_{AB} باشد.

۳. شروع کننده و پاسخ دهنده را وادار کنید که رشته‌های «چالش» خود را از مجموعه‌های متفاوتی انتخاب نمایند. مثلاً شروع کننده مجبور باشد اعداد زوج را انتخاب کند و پاسخ دهنده اعداد فرد را.

۴. پروتکل را در مقابل حملاتی که در اثر نشستهای موازی و همزمان امکان‌پذیر می‌شود، مقاوم کنید زیرا ممکن است اطلاعاتی که از یک نشست بدست می‌آید در دیگری قابل استفاده باشد.

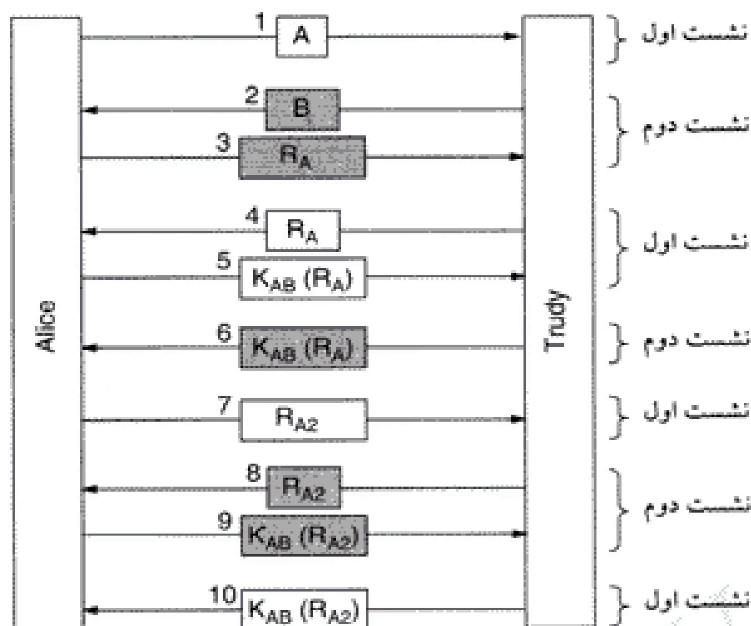
حتی اگر یکی از این چهار قاعده نقض شود، پروتکل به کرّات قابل شکستن خواهد بود. در پروتکل سه مرحله‌ای مثال قبل هر چهار قاعده نقض شده بود و بالطبع نتایج خطرناکی به بار خواهد آورد.

حال اجازه بدهید به شکل ۸-۳۲ بازگردیم و نگاهی دقیقتر به آن بیندازیم. آیا این پروتکل مطمئناً در معرض **«حملهٔ بازتاب»** (Reflection Attack) قرار نمی‌گیرد؟ بستگی دارد! قضیه کمی پیچیده است! ترودی می‌تواند این پروتکل را با استفاده از حملهٔ بازتاب شکست بدهد زیرا این امکان وجود دارد که او بتواند یک نشست دوم با باب ترتیب داده و او را به نحوی بفریبد تا به پرسشهای مورد نظر او پاسخ گوید. اگر آلیس را یک ماشین خودکار چندمنظوره فرض کنیم نه یک شخص حقیقی، با این ویژگی که بطور همزمان چندین نشست را می‌پذیرد، چه اتفاقی می‌افتد؟ بیایید بررسی کنیم در این وضعیت ترودی چه می‌تواند بکند.

برای آنکه متوجه شوید که حملهٔ ترودی به چه نحو خواهد بود به شکل ۸-۳۵ نگاه کنید. آلیس با اعلام مشخصهٔ شناسایی خود برای باب، کارش را شروع می‌کند. ترودی این پیام را راهزنی و متوقف کرده و با ارسال پیام ۲ خودش را به دروغ باب معرفی و نشست دومی را آغاز می‌کند. باز هم پیامهای نشست دوم به صورت خاکستری نشان داده شده‌اند. آلیس [در پاسخ به پیام ۲] پیامی بدین مضمون می‌فرستد: شما ادعا می‌کنید باب هستید؛ ثابت کنید! در اینجا به نظر می‌رسد که ترودی گیر افتاده چون نمی‌تواند ثابت کند باب است. ترودی چه کاری می‌تواند انجام بدهد؟

او به نشست اول بر می‌گردد؛ در آنجا نوبت اوست که رشتهٔ **«چالش»** خود را بفرستد، به همین دلیل R_A (ارسالی توسط آلیس) را می‌فرستد! آلیس در پاسخ، پیام ۵ را باز می‌گرداند. $K_{AB}(R_A)$ همان اطلاعاتی است که ترودی برای ادامهٔ نشست دوم بدان نیاز داشته است، لذا آنرا از طریق نشست دوم برای آلیس می‌فرستد. در اینجا ترودی در نشست دوم پاسخ موفقیت‌آمیزی را برای آلیس ارسال می‌کند. حال او می‌تواند نشست اول را لغو کرده و باقیماندهٔ مراحل نشست دوم را تکمیل کند و بدین ترتیب یک نشست موفق و مورد تایید با آلیس خواهد داشت. (نشست ۲)

ولیکن ترودی پلیدتر از این حرفهاست و می‌خواهد مردم‌آزاری خود را ادامه بدهد!!! به جای آنکه اعدادی قدیمی و بی‌ارزش را برای تکمیل نشست ۲ ارسال کند، منتظر می‌ماند که آلیس در ادامهٔ نشست اول رشتهٔ چالش خود یعنی R_{A2} را برایش بفرستد. اگرچه ترودی نمی‌داند که در پاسخ به آن چه باید برگرداند ولیکن باز هم از



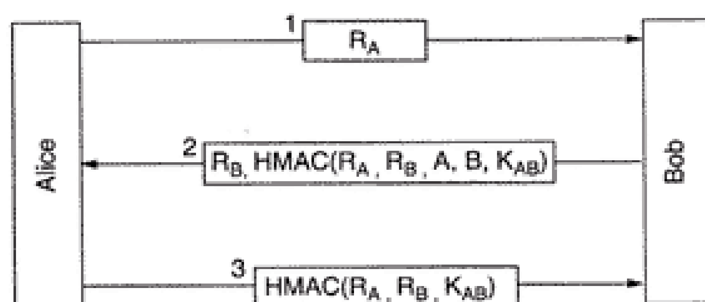
شکل ۸-۳۵. حملهٔ بازتاب بر علیه پروتکل شکل ۸-۳۲.

«حملهٔ بازتاب» بهره می‌گیرد و R_{A2} را در پیام هشتم ارسال می‌نماید. آلیس براحتی آن را رمز کرده و در پیام نهم بر می‌گرداند. ترودی مجدداً به نشست اول بازگشته و عدد بدست آمده از پیام نهم را در پیام ۱۰ برای آلیس پس می‌فرستد. در این لحظه ترودی دو نشست آماده و کامل با آلیس ترتیب داده است.

نتیجهٔ این حمله، با حمله به پروتکل سه‌مرحله‌ای شکل ۸-۳۴ متفاوت است. در اینجا ترودی دو ارتباط تائید شده و کامل با آلیس دارد در حالی که در مثال قبلی او فقط یک ارتباط تائید شده با باب برقرار کرده بود. در اینجا نیز اگر چهار قاعدهٔ اساسی در پروتکلهای احراز هویت که قبلاً عنوان کردیم رعایت می‌شد، چنین حمله‌ای ممکن نبود. تشریح کامل این نوع از حملات و چگونگی خنثی‌سازی آنها در مرجع (Bird et.al, 1993) آمده است. آنها نشان داده‌اند که می‌توان پروتکلهای متفاوتی را ایجاد کرد که صحّت عملکردشان قابل اثبات باشد. ساده‌ترین پروتکل از این‌گونه ، تا حدودی پیچیده است به همین دلیل ما در اینجا ردهٔ متفاوتی از پروتکلهای احراز هویت را معرفی خواهیم کرد.

پروتکل جدید احراز هویت در شکل ۸-۳۶ نشان داده شده است. (Bird et. al. 1993) در این پروتکل از HMAC[1] که در توضیح IPsec بدان اشاره کردیم استفاده شده است. آلیس در اولین پیام با ارسال R_A (به عنوان nonce یا همان رشته چالش) برای باب ، کارش را آغاز می‌کند. باب با انتخاب R_B برای خود، آن رادر قالب رشتهٔ درهم‌شدهٔ HMAC برای آلیس پس می‌فرستد. HMAC به گونه‌ای سازماندهی شده است که یک ساختمان داده شامل: R_A ارسالی توسط آلیس، R_B متعلق به باب و مشخصه‌های شناسایی هر دو و همچنین کلید مشترک آنهایعنی K_{AB} را در بر می‌گیرد. این ساختمان داده به روشی مثل SHA-1 درهم شده و در HMAC قرار می‌گیرد. وقتی آلیس پیام ۲ را دریافت می‌کند، R_A (که خودش در ابتدا انتخاب کرده بود)، R_B که به صورت آشکار برایش ارسال شده بود ، دو مشخصهٔ شناسایی (مشخصه خودش و باب) و کلید محرمانه را در اختیار دارد و بدین ترتیب می‌تواند بطور مستقل HMAC (یعنی رشتهٔ Hash) را برای این آیتمها محاسبه نماید. اگر HMAC محاسبه شده با

[1]. Hashed Message Authentication Code



شکل ۸-۳۶. احراز هویت با استفاده از روش HMAC.

HMAC ارسالی توسط باب معادل باشد، آلیس می‌تواند مطمئن باشد که در حال صحبت با باب است زیرا ترودی K_{AB} را نمی‌داند و بالطبع نمی‌تواند HMAC را به صورت جعلی و دروغین ساخته و آن را از طرف باب ارسال نماید. آلیس نیز در پاسخ، HMAC مربوط به سه آیتم R_A، R_B، K_{AB} را ارسال می‌کند. (باز هم ترودی نمی‌تواند این پیام را جعل کند، چون کلید K_{AB} را در اختیار ندارد.)

آیا ترودی می‌تواند به طریقی این پروتکل را شکست بدهد؟ خیر؛ چون او نمی‌تواند طبق وقایعی که در شکل ۸-۳۵ اتفاق افتاد، طرفین را وادار کند تا مقادیر دلخواه او را رمزنگاری یا رشتهٔ Hash را محاسبه نماید. هر دوی HMACها شامل مقادیری هستند که در کنترل ترودی نیست.

استفاده از HMAC تنها روش بهره‌گیری از ایدهٔ فوق نیست. اغلب به جای محاسبهٔ HMAC از روشی جایگزین استفاده می‌شود که در آن دنبالهٔ آیتمها به روش **«زنجیره‌سازی بلوکهای رمز»** رمز و ارسال می‌شوند.

۸-۷-۲ ایجاد کلید مشترک: مبادلهٔ کلید به روش «دیفی-هلمن»

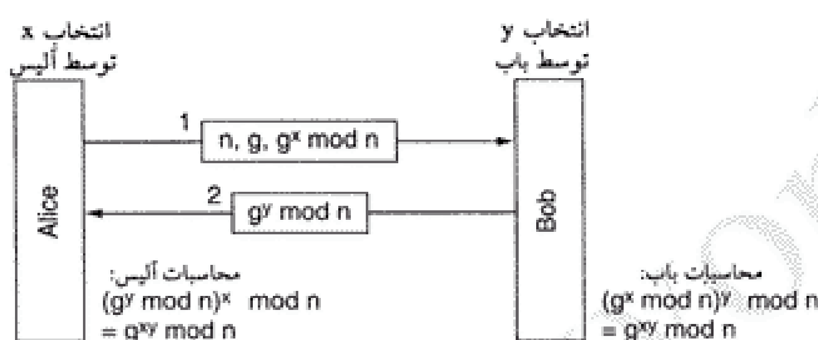
تا اینجا فرض کرده‌ایم که آلیس و باب بر روی یک کلید سرّی توافق کرده‌اند. حال فرض را بر آن بگذارید که آنها چنین کاری نکرده‌اند زیرا مثلاً هیچ مرکز جهانی و مورد وفاق PKI برای امضاء و توزیع گواهینامه‌های دیجیتالی وجود ندارد. در چنین حالتی چگونه این دو نفر می‌توانند یک کلید مشترک ایجاد کنند. یک راه آن است که آلیس از طریق تلفن با باب تماس گرفته و کلید مورد نظر خود را به او بگوید ولی ممکن است در همان ابتدای کار باب از آلیس سؤال کند که: «از کجا بدانم شما آلیس هستی و ترودی نیستی؟» آنها می‌توانند یک ملاقات ترتیب داده و هر کدام، گذرنامه یا گواهینامهٔ رانندگی خود و سه کارت اعتباری معتبر و مهم را همراه آورده و پس از تائید هویت یکدیگر، بر روی یک کلید مشترک توافق نمایند. برای مردم گرفتار، شاید تنظیم قرار ملاقات برای ماهها امکان‌پذیر نباشد. خوشبختانه راه حلّی برای توافق و ایجاد کلید سرّی بین افراد ناآشنا با یکدیگر وجود دارد که آنها می‌توانند در روز روشن و حتی وقتی ترودی قادر به استراق سمع و ضبط تمام پیامهاست، یک کلید سرّی ایجاد کنند، هر چند این موضوع ممکن است اندکی غیرقابل باور به نظر برسد.

پروتکلی که به افراد غریبه و ناآشنا با یکدیگر، اجازه می‌دهد یک کلید مشترک و سرّی ایجاد کنند، پروتکل **«مبادلهٔ کلید دیفی ـ هلمن»**[1] نام دارد و به صورت زیر عمل می‌کند: آلیس و باب می‌باید بر روی دو عدد بسیار بزرگ n و g توافق کنند که از این دو n عددی اول است. همچنین $(n-1)/2$ نیز عددی اوّل است و شرایط خاصی نیز بر روی g اعمال می‌شود. این دو عدد عمومی و غیرسرّی بوده و هر کسی می‌تواند آزادانه یک n و g انتخاب کرده و به دیگری اعلام کند. در اینجا آلیس یک عدد بزرگ x (مثلاً ۵۱۲ بیتی) انتخاب کرده و آن را به صورت سرّی نزد

۱. Diffie-Hellman key exchange; (Diffie and Hellman, 1976)

خود نگاه می‌دارد. به همین روش باب نیز یک عدد سرّی مثل y برای خود انتخاب می‌کند.

آلیس، طبق شکل ۸–۳۷، پروتکل مبادلهٔ کلید را با ارسال پیامی شامل آیتمهای ($g^x \bmod n$ و g و n) آغاز می‌کند. باب نیز در پاسخ، پیامی را ارسال می‌کند که در برگیرندهٔ $g^y \bmod n$ است. حال آلیس، عدد ارسالی توسط باب را در پیمانهٔ n به توان x می‌رساند تا حاصل $(g^y \bmod n)^x \bmod n$ به دست آید. باب نیز همین کار را به صورت $(g^x \bmod n)^y \bmod n$ انجام می‌دهد. طبق روابط حاکم بر نظریهٔ اعداد، هر دو نفر حاصل $g^{xy} \bmod n$ را بدست خواهند آورد. دقّت کنید! آلیس و باب به ناگاه صاحب یک کلید مشترک و سرّی با فرمول $g^{xy} \bmod n$ شده‌اند.

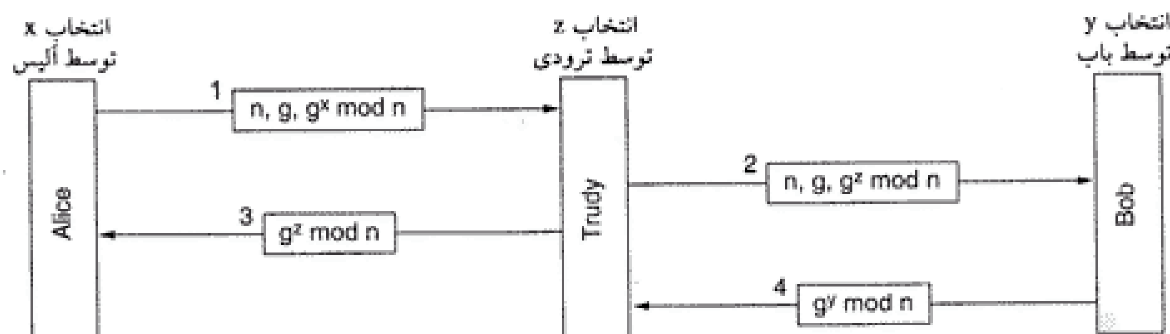


شکل ۸–۳۷. مبادلهٔ کلید بروش دیفی–هلمن.

البته در این میان ترودی هر دو پیام را می‌بیند بنابراین g و n را از پیام اول در اختیار دارد. اگر او بتواند x و y را محاسبه کند قادر خواهد بود تا کلید سرّی را تعیین نماید ولیکن مشکل اینجاست که با داشتن $g^x \bmod n$ نمی‌توان x را پیدا کرد زیرا هیچ الگوریتم عملی برای محاسبهٔ لگاریتم گسسته در پیمانهٔ اعداد اوّل بسیار بزرگ، تاکنون کشف نشده است.

برای آن که مثال بالا را بهتر تفهیم کنیم از اعداد $n=47$ و $g=3$ استفاده می‌نمائیم (این اعداد در عمل بسیار کوچک و غیرقابل استفاده‌اند). آلیس عدد $x=8$ و باب عدد $y=10$ را برای خود انتخاب می‌کند؛ هر دوی این اعداد سرّی نگه داشته می‌شوند. پیام آلیس به باب شامل آیتمهای (۲۸ و ۳ و ۴۷) خواهد بود چراکه حاصل ۳^۸ mod۴۷ معادل ۲۸ است. پیام باب به آلیس عدد ۱۷ است. آلیس مقدار ۱۷^۸ mod۴۷ را محاسبه می‌کند که ۴ بدست می‌آید. باب نیز حاصل ۲۸^۱۰ mod ۴۷ را بدست می‌آورد که باز هم ۴ است. در اینجا باب و آلیس هر دو کلید سرّی و مشترک ۴ را دارند که به صورت مستقل بدست آمده است. برای بدست آوردن کلید سرّی، ترودی مجبور است معادلهٔ $3^x \bmod 47 = 28$ را حل کند که فقط با جستجوی کامل [در محدودهٔ صفر تا n-1] امکان‌پذیر است و اگر n عددی بسیار بزرگ در حدّ چند صد بیت باشد، حل این معادله عملاً ممکن نخواهد بود. حتی اگر از ابرکامپیوترهای موازی استفاده شده باشد، تمام الگوریتمهای شناخته شدهٔ امروزی برای حلّ این معادله، نیاز به وقت بسیار زیادی دارند.

علیرغم زیبائی الگوریتم «دیفی ـ هلمن» ، مشکلی در آن وجود دارد: وقتی باب سه آیتم (۲۸ و ۳ و ۴۷) را دریافت می‌کند از کجا بداند که واقعاً از طرف آلیس پیشنهاد شده است نه از طرف ترودی؟ هیچ راهی برای این تشخیص وجود ندارد! متأسفانه ترودی می‌تواند به گونه‌ای که در شکل ۸–۳۸ نشان داده شده بطور همزمان آلیس و باب را فریب بدهد. در اینجا وقتی آلیس و باب اعداد x و y را برای خود انتخاب می‌کنند، ترودی نیز z را برای خود بر می‌گزیند. حال آلیس پیام اول را برای باب می‌فرستد ولیکن این پیام در میانهٔ راه توسط ترودی دریافت و متوقف می‌شود. ترودی g و n (که عمومی و غیرسرّی هستند) را به همراه $g^z \bmod n$ (به جای $g^x \bmod n$) برای



شکل ۸-۳۸. حملهٔ «گروه آتش‌نشان» (یا حملهٔ Man-In-The-Middle).

باب می‌فرستد. همچنین پیام سوم را به صورت $g^z \bmod n$ برای آلیس بر می‌گرداند. بعداً باب پیام چهارم یعنی $g^y \bmod n$ را برای آلیس می‌فرستد که آن هم توسط ترودی دریافت و حفظ می‌شود.

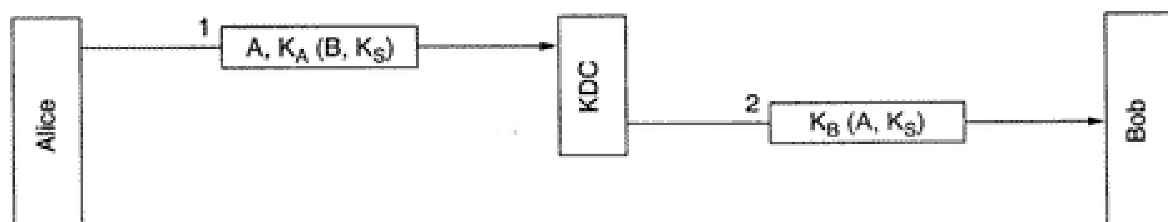
حال هر سه نفر محاسبات پیمانه‌ای (Modular Arithmetic) خود را آغاز می‌کنند. آلیس و ترودی کلید سرّی $g^{xz} \bmod n$ را محاسبه می‌کنند. از طرف دیگر باب و ترودی نیز $g^{yz} \bmod n$ را به عنوان کلید سرّی خود محاسبه می‌نمایند. آلیس در این خیال است که با باب صحبت می‌کند لذا یک کلید نشست با ترودی ایجاد کرده است. باب نیز به همین گونه فریب می‌خورد. هر پیامی که به صورت رمزنگاری شده توسط آلیس ارسال گردد ابتدا توسط ترودی دریافت، ذخیره و در صورت تمایل دستکاری شده و سپس برای باب ارسال می‌شود. در طرف مقابل نیز همین اتفاق می‌افتد. باب و آلیس با این تصور نادرست که کانالی امن ایجاد کرده‌اند، با یکدیگر تبادل اطلاعات می‌کنند در حالی که ترودی تمام پیامهای آنها را می‌بیند و احیاناً آنها را به میل خود تغییر می‌دهد. این حمله اصطلاحاً به نام **«حملهٔ گروه آتش‌نشان»** (Bucket Brigade Attack) مشهور است زیرا تقریباً به انتقال سطلهای آب از کامیون به محل آتش‌سوزی شباهت دارد که آتش‌نشانهای داوطلب در یک صف، سطلهای آب را دست به دست هدایت می‌کنند. این حمله همچنین به نام حملهٔ **mitm** (Man in The Middle Attack) شهرت دارد.

۸-۷-۳ احراز هویت توسط مرکز توزیع کلید (KDC)

توافق و تنظیم یک کلید مشترک و سرّی با افراد غریبه به روش فوق تقریباً عملی است ولیکن کامل و بی‌نقص نیست. حتی شاید (بدلیل ضعف امنیتی اشاره شده در بخش قبل و اشکالی که در ادامه بدان می‌پردازیم)، ارزش اجرایی نداشته باشد. شما برای آن که بتوانید با n نفر محاوره داشته باشید، طبعاً به n عدد کلید احتیاج خواهید داشت. برای افراد عادی، نگهداری و مدیریت کلیدها واقعاً سخت و پرمخاطره است، بالاخص اگر لازم باشد کلیدها بر روی یک کارت پلاستیکی ذخیره و تحویل شود.

راهکار متفاوتی که وجود دارد آنست که یک **«مرکز توزیع کلید»** مورد اعتماد و وفاق عموم (KDC)، معرفی شود. در این ساختار هر کاربر تنها یک کلید دارد که بین او و KDC مشترک است. احراز هویت و ایجاد کلید نشست، از طریق واسطهٔ KDC انجام می‌شود. در شکل ۸-۳۹ ساده‌ترین پروتکل احراز هویتِ مبتنی بر KDC نشان داده شده که شامل یک مرکز معتمدِ توزیع کلید و طرفین ارتباط است.

مبنای نظری این پروتکل بسیار ساده است: آلیس یک کلید نشست K_S را انتخاب کرده و به KDC اعلام می‌کند که تمایل دارد با استفاده از این کلید با باب محاوره نماید. این پیام با استفاده از کلید سرّی و مشترک بین آلیس و KDC که آن را K_A نامیده‌ایم رمز می‌شود. KDC این پیام را رمزگشایی کرده و مشخصهٔ شناسایی باب و کلید نشست را از درون آن استخراج می‌نماید. سپس KDC، پیام جدیدی را می‌سازد و در آن مشخصهٔ شناسایی آلیس و کلید نشست را قرار داده و پس از رمزنگاری برای باب می‌فرستد. رمزنگاری این پیام با کلید K_B یعنی کلید



شکل ۸-۳۹. اولین پروتکل احراز هویت بکمک KDC.

مشترک بین باب و KDC انجام می‌شود. وقتی باب این پیام را رمزگشایی کند، متوجه می‌شود که آلیس تمایل به محاوره با او دارد و کلید رمزی را که برای محاوره باید استفاده شود، بدست می‌آورد.

این روش احراز هویت بسیار ساده انجام می‌شود: KDC می‌داند که پیام قاعدتاً از طرف آلیس آمده است چراکه هیچکسی قادر به رمزنگاری این پیام با کلید سرّی آلیس نیست؛ (چون کلید آلیس را ندارد). به روش مشابه، باب نیز می‌داند که پیام ۲ از طرف KDC که مورد اعتماد اوست صادر شده چون هیچکس کلید سرّی او را در اختیار ندارد (مگر KDC).

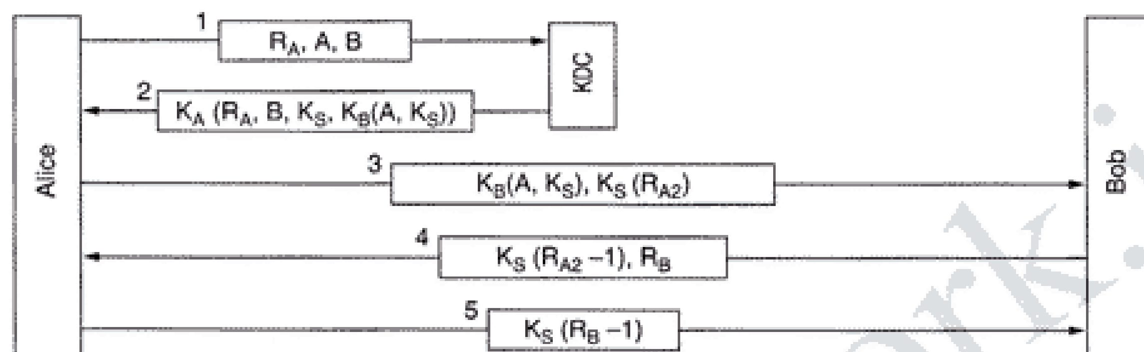
متأسفانه این پروتکل دارای یک اشکال جدّی است. به سناریوی زیر دقت کنید. ترودی به مقداری پول احتیاج دارد لذا به گونه‌ای برنامه‌ریزی می‌کند که خدماتی متعارف برای آلیس انجام بدهد و سپس به استخدام موقت او در می‌آید. پس از انجام کار، ترودی مؤدبانه از آلیس خواهش می‌کند تا حقوقش را به حساب بانکی او واریز نماید. آلیس با بانکدار خود یعنی باب، یک کلید نشست ایجاد کرده و سپس از باب تقاضا می‌کند که پول درخواستی را از حساب او به حساب ترودی واریز نماید.

در این میان، ترودی به راه و روش قبلی خود یعنی جاسوسی و پرسه زدن در شبکه برمی‌گردد. او پیام شماره ۲ در شکل ۸-۳۹ و همچنین پیام تقاضای انتقال پول [که در شکل نیامده] را استراق سمع نموده و در جایی کپی می‌کند. ترودی بعداً این دو پیام را بار دیگر برای باب تکرار می‌کند. باب این پیامها را دریافت کرده و با خود می‌اندیشد که آلیس ترودی را برای بار دیگر به استخدام خود در آورده است لذا همان مقدار پول را از حساب بانکی آلیس به حساب ترودی واریز می‌کند. پس از دریافت پنجاهمین پیام، باب از دفترش بیرون می‌آید تا ترودی را پیدا کرده و برای توسعهٔ کار تجارت به او پیشنهاد وام بدهد!!! به این نوع حمله اصطلاحاً **«حملهٔ تکرار»** (Replay Attack) گفته می‌شود.

چندین راه حل برای جلوگیری از حملهٔ تکرار وجود دارد. اولین راه حل آن است که هر پیام دارای **«مُهر زمان»** (Timestamp) باشد. در این صورت هرگاه کسی یک پیام قدیمی دریافت کرد می‌تواند براحتی آنرا حذف کند. یکی از مشکلات این روش آن است که نمی‌توان در یک شبکه، همهٔ ساعتها را بدقت با هم تنظیم کرد لذا باید به گونه‌ای برنامه‌ریزی نمود که مهر زمان در یک بازهٔ مشخص [مثلاً پانزده دقیقه] اعتبار داشته باشد و بدین ترتیب ترودی در این بازهٔ زمانی مهلت خواهد داشت تا پیامهای مورد نظر خود را تکرار کرده و به منظور خود برسد.

راه حل دوم آن است که در هر پیام یک عدد یا رشتهٔ تصادفی چالش (nonce) گذاشته شود و پیامهایی که این عدد یا رشته در آنها تکراری است حذف شوند. مشکل این روش آن است که تمام این اعداد یا رشته‌ها باید برای همیشه نگه‌داری شوند مبادا ترودی پیامی مربوط به پنج سال قبل را به صورت تکراری بفرستد. همچنین اگر ماشینی درهم بشکند و فهرست این اعداد یا رشته‌ها از دست برود در معرض «حملهٔ تکرار» قرار خواهد گرفت. می‌توان از ترکیب «مُهر زمان» و «الصاق عدد یا رشتهٔ تصادفی چالش» (nonce) در پیامها استفاده کرد. بدین ترتیب طول مدتی که باید این اعداد و رشته‌ها حفظ شوند کاهش پیدا می‌کند ولیکن پروتکل پیچیده‌تر خواهد شد.

راهکاری پیچیده‌تر برای آنکه طرفین بتوانند خودشان به کمک KDC، یکدیگر را احراز هویت کنند آنست که از یک پروتکل «چالش-پاسخ» (Challenge/Response) چند مرحله‌ای استفاده شود. مثالی از این نوع، **«پروتکل احراز هویت نیدهام ـ شرودر»** (Needham-Schroeder, 1978) است که گونه‌ای از آنرا در شکل ۸–۴۰ می‌بینید.



شکل ۸–۴۰. پروتکل احراز هویت «نیدهام-شرودر».

این پروتکل بدین نحو آغاز می‌شود که آلیس به KDC اعلام می‌کند تمایل به محاوره با باب دارد. این پیام شامل یک عدد تصادفی بزرگِ R_A است. KDC پیام ۲ را بر می‌گرداند که حاوی: عدد تصادفی آلیس (یعنی R_A)، کلید نشست (یعنی K_S)، مشخصهٔ شناسایی باب (یعنی B) و یک **«بلیط»** (یعنی $K_B(A,K_S)$) است؛ این بلیط باید دست نخورده برای باب ارسال شود. هدف از R_A آن است که از غیرتکراری و غیرجعلی بودن پیام ۲ اطمینان حاصل شود. مشخصهٔ شناسایی باب بدان جهت درون این پیام جاسازی شده که اگر تـرودی در میانهٔ راه، پیام شمارهٔ یک را دستکاری نموده و مشخصهٔ خود را با مشخصه باب B عوض کرده باشد، قضیه آشکار شود چراکه در غیر این صورت KDC بلیط را به جای K_B با K_T (کلید ترودی) رمزنگاری کرده و بر می‌گرداند. «بلیط» ابتدا با کلید K_B رمزنگاری شده و سپس درون پیامی قرار می‌گیرد که بار دیگر با کلید K_A رمز خواهد شد تا ترودی بهیچوجه نتواند محتوای این پیام (پیام ۲) را دستکاری کرده و به آلیس تحویل بدهد.

حال آلیس بلیط خود ($K_B(A,K_S)$) را برای باب می‌فرستد و به همراه آن یک عدد تصادفی بزرگ R_{A2} را نیز با کلید نشست K_S رمز کرده و ارسال می‌کند. در پیام چهارم باب حاصل رمزنگاری $K_S(R_{A2}$-1) را برای آلیس باز پس می‌فرستد تا ثابت کند که آلیس واقعاً با باب صحبت می‌کند. برگرداندن $K_S(R_{A2})$ عملی نخواهد بود زیرا ترودی می‌تواند آن را از پیام سوم استراق سمع کند و به صورت جعلی به آلیس بازگرداند.

پس از دریافت پیام چهارم آلیس متقاعد می‌شود که با باب واقعی صحبت می‌کند و هیچ پیام جعلی و تکراری دریافت نکرده چراکه او R_{A2} را در چند میلی‌ثانیه قبل و به صورت تصادفی تولید نموده است. مقصود از ارسال پیام پنجم آن است که آلیس، باب را متقاعد کند که واقعاً آلیس است و پیام سوم، جعلی و تکراری نبوده است. چون هر یک از طرفین رشته‌های «چالش» (Challenge) و «پاسخ» (Response) را مبادله می‌کنند لذا امکان «حملهٔ تکرار» (Replay) منتفی است. [در این ساختار، KDC نیز نقش کوتاه ولی بسیار موثری ایفاء می‌کند.]

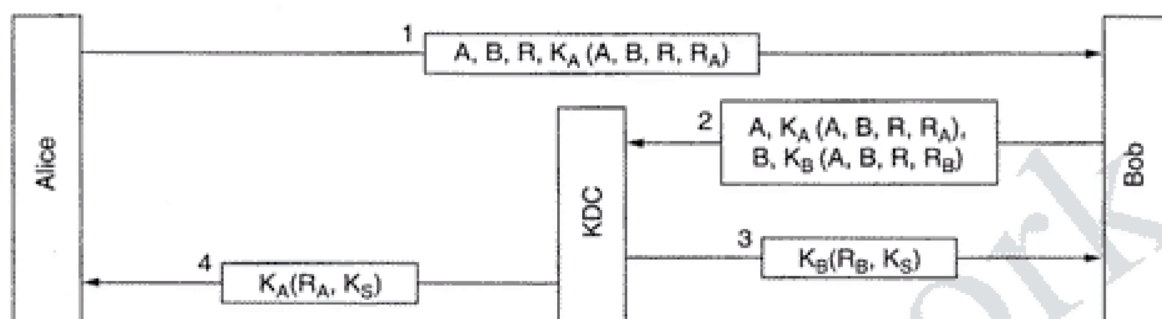
اگرچه این پروتکل بسیار محکم به نظر می‌رسد ولیکن یک ضعف بسیار جزیی دارد. اگر ترودی بتواند با برنامه‌ریزی یک کلید نشستِ قدیمی که در گذشته از آن استفاده شده را بنحوی بدست بیاورد[1] می‌تواند به صورت

---

۱. منظور از بدست آوردن کلید نشست قدیمی، شکستن رمز نیست بلکه سرقت کلیدی است که مثلاً بدلیل قدیمی بودن حذف نشده و به نحوی لو رفته است. -م.

جعلی از مرحلهٔ سوم شروع کرده و نشست جدیدی را با باب شروع کند و او را متقاعد نماید که آلیس است. در اینجا ترودی می‌تواند بدون هیچ زحمت اضافه‌ای، مثلاً حساب بانکی آلیس را چپاول کند.

برای رفع این اشکال «نیدهام و شرودر» پروتکلی را تدوین کردند. (۱۹۸۷) جالب آن که در همان شماره از مجلهٔ علمی که آنها پروتکل خود را به چاپ رساندند دو نفر دیگر به نامهای «اتوی» و «ریس» (Otway & Rees) نیز پروتکلی را تدوین و عرضه کرده بودند که این مشکل را به روشی ساده‌تر و کوتاه‌تر حل می‌کرد. شکل ۸-۴۱ پروتکل «اتوی-ریس» را با تغییر جزیی نشان می‌دهد.



شکل ۸-۴۱. پروتکل احراز هویت «اتوی-ریس».

در پروتکل اتوی-ریس، آلیس کار خود را با تولید دو عدد تصادفی بزرگ آغاز می‌کند: R که به عنوان یک شناسهٔ مشترک بکار می‌رود و R_A که آلیس از آن به عنوان رشتهٔ «چالش» (Challenge) استفاده می‌نماید. وقتی باب پیام اول را [طبق شکل] دریافت می‌کند، با استفاده از بخش رمزنگاری شدهٔ پیام آلیس، پیام جدیدی را ساخته و برای KDC می‌فرستد. در این پیام، باب آیتمی مشابه با بخش رمزنگاری شدهٔ پیام اول تولید و به آن می‌افزاید.[1] K_A و K_B کلیدهای محرمانهٔ آلیس و باب هستند و آیتمهای A و B و R و R_A یا R_B با آنها رمز می‌شوند.

حال از آنجایی که KDC کلید رمز هر دو نفر را در اختیار دارد، با دریافت این پیام ابتدا بررسی می‌کند که آیا R بدست آمده از دو پیام رمزنگاری شده یکی است یا خیر، زیرا ممکن است به دلیل دستکاری در پیام اول توسط ترودی، با هم مغایرت داشته باشند. اگر هر دو R با هم یکی بودند KDC متقاعد می‌شود که پیام ارسالی از باب [که در حقیقت تقاضای احراز هویت به حساب می‌آید] معتبر است. سپس برای هر دو نفر یک کلید نشست تولید نموده و آن را یکبار به همراه R_A، با کلید آلیس رمز کرده و برای آلیس می‌فرستد و بار دیگر به همراه R_B با کلید باب رمز کرده و برای باب ارسال می‌دارد. هر کدام از این پیامها (پیام ۳ و ۴) شامل عدد تصادفی تولید شده توسط گیرندهٔ آن پیام است که ثابت می‌کند واقعاً KDC آن را تولید کرده و به صورت جعلی یا تکراری توسط ترودی ارسال نشده است. در این لحظه آلیس و باب دارای کلید مشترکی هستند که توسط KDC پیشنهاد شده و می‌توانند با اطمینان محاورهٔ خود را آغاز نمایند. در اولین پیامی که بین آنها ردّ و بدل می‌شود آنها می‌توانند بررسی کنند که آیا دیگری K_S مشابهی دارد یا خیر. [چراکه در غیر این صورت پیامهای ارسالی از رمز خارج نخواهد شد.]؛ بدین ترتیب مراحل احراز هویت تکمیل می‌شود.

۸-۷-٤ احراز هویت با استفاده از Kerberos

یک پروتکل احراز هویت که در بسیاری از سیستمهای واقعی (مثل ویندوز ۲۰۰۰) بکار گرفته می‌شود، Kerberos است که براساس گونه‌ای از پروتکل «نیدهام ـ شرودر» بنا نهاده شده است. نام Kerberos برگرفته از یک اسطورهٔ

۱. یعنی پیام دوم شامل K_A(A,B,R,R_A) و A است که توسط آلیس ارسال شده و در آن دخل و تصرفی نمی‌شود و همچنین شامل K_B(A,B,R,R_B) و B است که خودش آنرا می‌سازد. -م.

یونانی است که در آن یک سگ چند سر از درب دوزخ نگهبانی می‌کند (که مثلاً نگذارد کسی از آن خارج شود)!! Kerberos در دانشگاه MIT طراحی شده و به کاربران اجازهٔ دسترسی مطمئن به منابع شبکه را می‌دهد. مهمترین تفاوت آن با پروتکل «نیدهام ـ شرودر» آن است که فرض شده ساعت تمام ایستگاههای شبکه دقیقاً با هم تنظیم شده‌اند. این پروتکل مراحل پیاده‌سازی متعددی را پشت سر گذاشته است. نسخهٔ V4 آن کاربرد بسیار گسترده‌ای در صنعت دارد به همین دلیل، آن را توضیح می‌دهیم. سپس چند کلمه‌ای در خصوص نسخهٔ بعدی آن V5 سخن خواهیم گفت. برای اطلاعات بیشتر به مرجع (Steiner et. al.; 1988) مراجعه کنید.

Kerberos به غیر از آلیس به عنوان ماشین مشتری، با سه ماشین سرویس دهندهٔ دیگر سر و کار دارد:

۱. **سرویس دهندهٔ احراز هویت یا** *AS* (Authentication Server): این سرویس دهنده کاربران را در حین ورود به سیستم (Login) بازرسی می‌نماید.

۲. **سرویس دهندهٔ صدور بلیط یا** *TGS* (Ticket Granting Server): این سرویس دهنده بلیطهای تائید هویت صادر می‌کند.

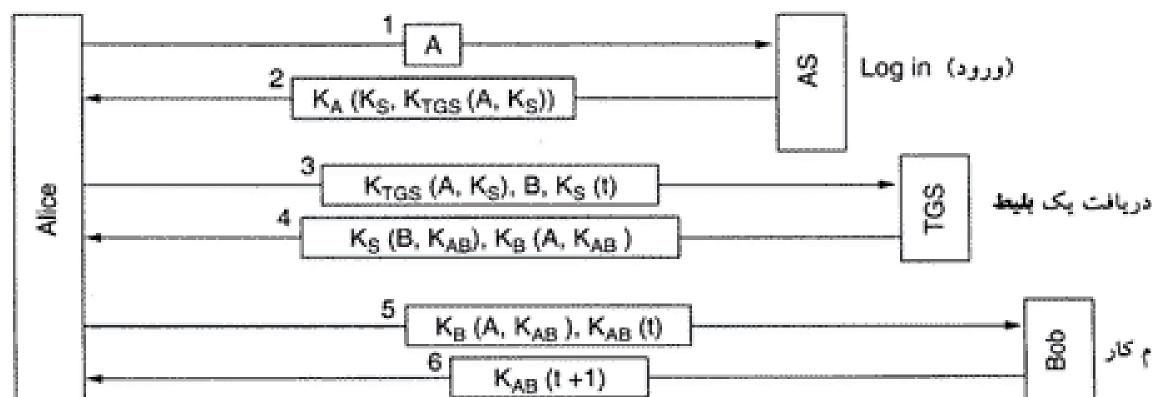
۳. **سرویس دهندهٔ باب** (یا هر سرویس دهنده‌ای که خدماتی را ارائه می‌کند): این سرویس دهنده کاری را که آلیس از او می‌خواهد انجام می‌دهد.

سرویس دهندهٔ AS شبیه به KDC عمل می‌کند که در آن کلیدهای سرّی تمام کاربران شبکه نگهداری می‌شود. کار سرویس دهندهٔ TGS آن است که بلیطهای معتبری را برای کاربران صادر کند تا سرویس‌دهنده‌های واقعی در شبکه متقاعد شوند که حامل بلیط، واقعاً همان کسی است که ادعا می‌کند.

برای شروع یک نشست، آلیس در جلوی هر یک از ایستگاههای عمومی شبکه نشسته و نام خود را درج (تایپ) می‌کند. ایستگاه (Workstation) نام او را به صورت آشکار برای سرویس دهندهٔ AS می‌فرستد. مراحل احراز هویت در شکل ۸-۴۲ نشان داده شده است. آنچه که در پاسخ باز برمی‌گردد یک «کلید نشست» (یعنی K_S) و یک «بلیط» یعنی $K_{TGS}(A,K_S)$ است که باید بعداً به سرویس دهندهٔ TGS تحویل داده شود. این دو آیتم درون یک پیام جاسازی شده و با استفاده از کلید سرّی آلیس رمز می‌شود فلذا هیچکس جز آلیس نمی‌تواند آن را رمزگشایی کند. وقتی پیام ۲ دریافت شود، ایستگاه از آلیس کلمهٔ عبور او را سؤال می‌کند. سپس از کلمهٔ عبور، کلید رمز آلیس یعنی K_A تولید می‌شود تا پیام ۲ رمزگشایی شده و کلید نشست و بلیط TGS از درون آن استخراج شود. در این لحظه ایستگاه فوراً کلمهٔ عبور آلیس را از حافظه پاک می‌کند تا بیش از چند میلی ثانیه در حافظهٔ ایستگاه باقی نماند. اگر ترودی سعی کند با نام آلیس وارد شود (Login کند)، کلمهٔ عبوری که درج می‌کند غلط خواهد بود و ایستگاه این موضوع را براحتی کشف خواهد کرد چراکه بدنهٔ اصلی پیام فقط با کلید آلیس از رمز خارج خواهد شد و بدون این کلید پیام دوم غیرقابل استفاده و نامفهوم خواهد بود.

پس از آن که آلیس به شبکه وارد شد ممکن است بخواهد با سرویس دهندهٔ فایلِ باب، ارتباط برقرار کند. در اینجا ایستگاه پیام ۳ را به سرویس دهندهٔ TGS می‌فرستد و از او می‌خواهد تا بلیطی برای استفاده از سرویس دهندهٔ باب صادر نماید. نکتهٔ اساسی در این تقاضا $K_{TGS}(A,K_S)$ است که با کلید سرّی سرویس‌دهندهٔ TGS رمزنگاری شده و محتوای آن ثابت می‌کند که فرستندهٔ بلیط واقعاً آلیس است. سرویس دهندهٔ TGS با ایجاد یک کلید نشست K_{AB} برای آلیس، به او امکان استفاده از آن را در محاوره با باب می‌دهد. دو نسخه از این کلید برگشت داده می‌شود: اولی با کلید نشست K_S رمز شده و آلیس می‌تواند آن را استخراج کرده و بخواند. نسخهٔ دوم با کلید سرّی باب یعنی K_B رمز می‌شود که هیچکس جز باب نمی‌تواند آن را بخواند.

ترودی می‌تواند پیام ۳ را دریافت و کپی کند و تلاش نماید تا آن را مجدداً بکار بگیرد ولیکن تلاش او با وجود مُهر زمان t که رمزنگاری نیز شده است، بی‌ثمر خواهد ماند. ترودی نمی‌تواند مهر زمانِ درج شده در پیام ۳ را



شکل ۸-۴۲. عملکرد سیستم Kerberos V4 .

عوض کرده و زمان جدیدی در آن درج نماید زیرا از کلید نشست K_S که آلیس بکمک آن با TGS محاوره می‌کند، بی‌اطلاع است. حتی اگر ترودی بلافاصله پیام ۳ را استراق سمع و تکرار کند، کپی پیام چهارم را بدست خواهد آورد که قادر نخواهد بود آن را فوراً رمزگشایی کند. (نه قسمت اول آن را که با K_S رمز شده و نه قسمت دوم را که با K_B رمز شده است.)

حال آلیس می‌تواند K_{AB} را برای باب فرستاده و یک نشست با او ترتیب بدهد. این مرحله نیز به همراه مهر زمان (Timestamp) خواهد بود. پاسخی که آلیس از باب دریافت می‌کند ثابت خواهد کرد که واقعاً با باب محاوره می‌کند نه ترودی. [بدین نحو هویت باب تائید می‌شود.]

پس از این چند مرحله مبادلهٔ پیام (جمعاً ۶ پیام) آلیس می‌تواند در پوشش کلید رمز K_{AB} با باب تبادل اطلاعات داشته باشد. هر گاه او تصمیم بگیرد که با یک سرویس دهندهٔ دیگر محاوره نماید (مثلاً سرویس دهندهٔ کارول) از پیام سوم شروع می‌کند با این تفاوت که به جای مشخصهٔ شناسایی B در این پیام، C را درج می‌کند. TGS سریعاً برای او بلیطی صادر می‌کند که به جای K_B با K_C رمز شده است و آلیس می‌تواند آن را برای کارول بفرستد و کارول نیز آن را به عنوان یک سند معتبر که از طرف آلیس ارسال شده می‌پذیرد.

نکتهٔ ارزشمند در این ساختار آنست که آلیس می‌تواند به تمام سرویس‌دهنده‌های موجود در شبکه به روشی مطمئن دسترسی داشته باشد و به هیچ وجه کلمهٔ عبور او بر روی شبکه منتقل نخواهد شد. در حقیقت کلمهٔ عبور او فقط برای چند میلی ثانیه درون ایستگاهی که او در کنار آن قرار دارد، ذخیره می‌شود. به این نکته دقت کنید که هر سرویس دهنده بعداً احراز هویت خاص خود را انجام می‌دهد یعنی وقتی آلیس بلیط خود را به سرویس‌دهندهٔ باب عرضه می‌کند، تنها کاری که این بلیط انجام می‌دهد آن است که برای باب ثابت کند که او واقعاً آلیس است. حال کارهایی که آلیس مجاز به انجام آن است صرفاً توسط سرویس دهندهٔ باب تعیین می‌شود.

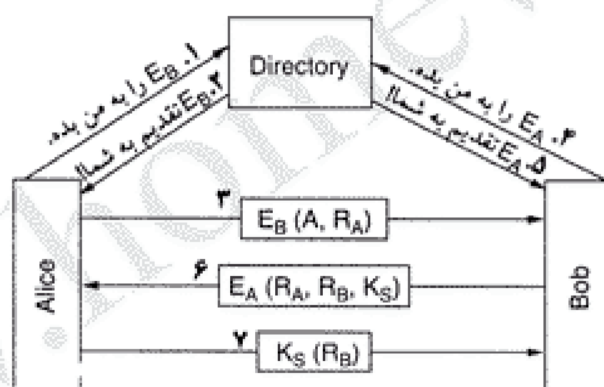
از آنجایی که طراحان Kerberos توقع نداشتند که کل دنیا به یک سرویس دهندهٔ واحد و جهانی احراز هویت اعتماد کنند، لذا این امکان را فراهم کردند که تعدادی ناحیهٔ مستقل وجود داشته باشد و هر ناحیه از یک AS و TGS خاصّ خود استفاده کند. آلیس برای آن که بتواند بلیطی را برای یک سرویس دهنده در ناحیه‌ای دور به دست بیاورد، از سرویس‌دهندهٔ TGS خودش می‌خواهد که بلیطی برایش صادر کند که موردِ پذیرش TGS ناحیهٔ راه دور باشد. اگر TGS ناحیهٔ دور در TGS محلی ثبت شده باشد (به همان روشی که سرویس دهنده‌های محلی ثبت می‌شوند)، TGS محلی به آلیس بلیطی می‌دهد که نزد TGS دیگر معتبر است. او می‌تواند با این بلیط کار دلخواهش را در آن TGS انجام بدهد مثلاً می‌تواند از آن TGS بلیط دسترسی به یک سرویس‌دهندهٔ دیگر در آن ناحیه را دریافت نماید. بهرحال برای آن که طرفین در دو ناحیهٔ مختلف بتوانند کارشان را انجام بدهند باید به TGS

یکدیگر اعتماد داشته باشند.

نسخهٔ V5 از Kerberos، مفصلتر از نسخهٔ V4 است و زحمت و سربار زیادتری دارد. همچنین برای توصیف انواع داده (Data Types) از استاندارد OSI ASN.1 (Abstract Syntax Notation 1) استفاده کرده است و در ساختار پروتکل نیز تغییرات اندکی ایجاد شده است. به علاوه، طول عمر بلیطها زیادتر شده و اجازه داده شده بلیطها تجدید شوند. به علاوه پروتکل V5، لااقل در تئوری به DES وابسته نیست (در حالی که V_4 این‌گونه است) و از وجود نواحی مختلف و تفویض صدور کلید به سرویس‌دهنده‌های این نواحی حمایت می‌نماید.

۸-۷-۵ احراز هویت با استفاده از رمزنگاری با کلید عمومی

می‌توان عملیات احراز هویت را با استفاده از رمزنگاری با کلید عمومی انجام داد. برای شروع، آلیس نیاز دارد که کلید عمومی باب را بدست بیاورد. اگر PKI[1] (مرکز توزیع کلید عمومی) با ساختار «سرویس‌دهندهٔ دایرکتوری» در جایی از شبکه موجود باشد و گواهینامه‌های کلید عمومی را تحویل بدهد، آلیس می‌تواند به نحوی که در پیام ۱ از شکل ۸-۴۳ نشان داده شده است، از این سرویس‌دهنده در مورد باب سؤال نماید. در پاسخ، پیام ۲ که حاوی گواهینامه X.509 و کلید عمومی باب است، بازمی‌گردد. پس از آن که آلیس صَحتِ امضای باب را بررسی و تائید کرد، برای او پیامی می‌فرستد که در آن مشخصهٔ شناسایی خودش و یک عدد تصادفی (R_A یا Nonce) با کلید عمومی باب رمز شده است.



شکل ۸-۴۳. احراز هویت متقابل با استفاده از رمزنگاری کلید عمومی.

وقتی باب این پیام را دریافت می‌کند، نمی‌داند که آیا این پیام واقعاً از طرف آلیس ارسال شده یا ترودی، ولی به همان روش عمل کرده و از «سرویس‌دهندهٔ دایرکتوری» در مورد کلید عمومی آلیس سؤال می‌کند (در پیام ۴) و پاسخ آن را (در پیام ۵) به دست می‌آورد. سپس در پیام ششم R_A متعلق به آلیس، عدد تصادفی خودش (R_B) و یک کلید نشست K_S پیشنهادی را برای آلیس می‌فرستد. وقتی آلیس پیام ششم را دریافت می‌کند آن را با استفاده از کلید خصوصی خودش رمزگشایی کرده و R_A خود را درون آن می‌بیند که به او در خصوص هویت باب اطمینان قلبی می‌بخشد. این پیام قطعاً باید از طرف باب آمده باشد چراکه ترودی هیچ راهی برای تعیین R_A ندارد. همچنین این پیام تکراری نبوده و جدید است چراکه باب R_A را [که به صورت تصادفی و غیرتکراری تولید شده است] بازگردانده است. آلیس با برگرداندن پیام هفتم، بر روی کلید نشست توافق می‌کند. وقتی باب، R_B ارسالی خود را که با کلید نشست رمزنگاری و بازگردانده شده، در پیام هفتم مشاهده می‌کند، متوجه می‌شود که آلیس پیام ششم را گرفته و R_A را بررسی کرده است.

۱. بخش ۸-۵ را ببینید. -م.

چگونه ترودی می‌تواند این پروتکل را درهم بریزد و در آن رسوخ کند؟ او می‌تواند پیام شماره ۳ را به صورت جعلی تولید و از طرف آلیس برای باب بفرستد [چرا که کلید عمومی او یعنی E_B را همه و از جمله ترودی می‌دانند] ولی وقتی آلیس در پیام ششم R_A را مشاهده می‌کند متوجه می‌شود که چنین عددی را او نفرستاده و به همین دلیل مراحل کار را ادامه نخواهد داد. ترودی قادر نخواهد بود پیام هفتم را به صورت جعلی تولید کرده و برای باب بفرستد، چون R_B و K_S را نمی‌داند و به هیچ وجه قادر نخواهد بود آنها را بدون کلید خصوصی آلیس محاسبه و تعیین نماید. لذا او هیچ اقبالی نخواهد داشت.

۸ـ۸ امنیت نامه‌های الکترونیکی

وقتی یک پیام توسط پست الکترونیکی بین دو سایت راه دور مبادله می شود، بطور معمول آن نامه در طی مسیر خود از دهها ماشین میانی عبور خواهد کرد. هر یک از این ماشینها قادرند آن را بخوانند یا برای استفاده‌های بعدی ذخیره کنند. برخلاف آنچه که بسیاری از مردم می‌اندیشند، حریم خصوصی عملاً وجود ندارد ولیکن علیرغم این بسیاری از افراد علاقمندند نامه‌هایی را که ارسال می‌کنند فقط گیرنده مورد نظر بخواند نه هیچکس دیگر؛ نه رئیس آنها و نه حتی حکومت. این احساس نیاز بسیاری از گروهها و افراد را ترغیب کرد که اصول رمزنگاری را که تا اینجا بررسی کردیم، بر روی نامه‌های الکترونیکی اعمال کرده و یک سیستم امن پست الکترونیکی به وجود بیاورند. در بخش بعدی، PGP را مطالعه خواهیم کرد که یک سیستم پست الکترونیکی امن و بسیار رائج است، سپس به اختصار به دو روش PEM و S/MIME نگاهی خواهیم انداخت. برای دسترسی به اطلاعات غنی‌تر در مورد سیستمهای امن پست الکترونیکی، مرجع (Kaufman et. al. 2002; Schneier, 1995) را ملاحظه نمائید.

۸ـ۸ـ۱ PGP (Pretty Good Privacy)

PGP ، بـه عـنوان اولین مثال از سیستم پست الکترونیکی امن، زائیدهٔ تفکر شخصی به نام «زیمرمَن» (Phil Zimmermann,1995) بود. شعار زیمرمن که یکی از طرفداران حفظ حریم خصوصی افراد است، این بود که: «اگر ایجاد حریم خصوصی و حفظ اسرار مردم قانون‌شکنی است آنگاه فقط حریم خصوصی قانون‌شکنان حفظ خواهد شد.» PGP یک بستهٔ نرم‌افزاری کامل برای امنیت نامه‌های الکترونیکی است که در سال ۱۹۹۱ منتشر شد و با یک ساختار بسیار سادهٔ کاربری، تمام امکانات ذیل شامل **«تدوین نامه‌های خصوصی»** (رمزشده)، **«احراز هویت»**، **«امضای دیجیتالی»** و حتی **«فشرده‌سازی اطلاعات»** را به صورت یکجا عرضه کرده است. به علاوه، بستهٔ نرم‌افزاری بهمراه کدهای برنامه، بصورت رایگان از طریق اینترنت در دسترس عموم قرار می‌گیرد. امروزه به دلیل کیفیت بالا، عدم هزینه (قیمت صفر) و وجود نسخه‌های متفاوت بر روی یونیکس، لینوکس، ویندوز و سیستم عامل مکینتاش (Mac OS)، از آن به صورت گسترده‌ای استفاده می‌شود.

PGP داده‌ها (محتویات نامه) را با استفاده از روشی به نام IDEA[۱] که مبتنی بر رمزنگاری بلوکی است رمز می‌کند و در آن از کلیدهای ۱۲۸ بیتی و متقارن بهره می‌گیرد. این روش رمزنگاری در سوئیس زمانی ابداع شد که ضعف و اشکالات DES ارزش آن را خدشه‌دار کرده بود ولی هنوز AES اختراع و معرفی نشده بود. IDEA شبیه به DES و AES است: در چندین دور (Round) بیتهای داده را با هم ترکیب می‌کند ولیکن جزئیات توابع ترکیب بیتها، در مقایسه با DES و AES متفاوت است. در PGP، برای مدیریت کلیدها از RSA و برای بررسی صحت داده‌ها از MD5 استفاده شده که این روشها را قبلاً معرفی کردیم.

از اولین روز معرفی PGP ، جنگ و جدل وسیعی پیرامون آن در گرفت، (Levy, 1993) از آنجایی که زیمرمن

۱. International Data Encryption Algorithm

هیچ اقدامی برای آن که افراد را از توزیع PGP بر روی اینترنت منع کند انجام نداده بود و هر کسی در هر نقطهٔ دنیا می‌توانست بدان دسترسی داشته باشد، دولت ایالات متحده او را متهم ساخت که قوانین «منع صدور ابزارهای استراتژیک» را نقض کرده است. بازجویی دولت آمریکا از زیمرمن پنج سال طول کشید ولی نهایتاً پرونده مختومه شد؛ شاید به دو دلیل: اول آن که زیمرمن شخصاً PGP را بر روی اینترنت نگذاشته بود و به همین دلیل وکلای او ادعا کردند که او هیچ چیزی را صادر نکرده است. ثانیاً دولت به این نتیجه رسید که پیروزی در این دادگاه و محکومیت زیمرمن بدان معنا تلقی می‌شود که یک وب‌سایت که برنامه‌ای قابل دریافت (Downloadable) در خصوص امنیت ارائه می‌دهد در شمول قانون منع فروش تجهیزات استراتژیک مثل تانک، زیردریایی، هواپیماهای نظامی و موشکهای هسته‌ای قرار گرفته است. سالها تبلیغات منفی، احتمالاً هیچ کمکی به آنها نکرده بود و قرار گرفتن سایتهای وب در شمول تجهیزات جنگی یک تبلیغ منفی بزرگ برای دولت بود.

دولت قرار دادن کدهای یک برنامه بر روی وب‌سایت را در شمول صادرات غیرقانونی قرار داد و بدین ترتیب زیمرمن را به مدت پنج سال گرفتار دادگاه کرد در حالیکه اگر کسی کدهای برنامهٔ PGP را به زبان C در یک کتاب تدوین (و آن را با حروف بزرگ و قابل اسکن چاپ می‌نمود) و سپس آن کتاب را صادر می‌کرد توسط دولت فقط جریمه می‌شد چرا که کتاب در ردهٔ تجهیزات جنگی قرار نمی‌گیرد. به هر حال لااقل برای عموسام قدرت شمشیر از قلم بیشتر است.

یکی دیگر از مشکلات حقوقی PGP ، مسئلهٔ نقض مالکیت امتیاز بود. شرکتی که امتیاز RSA را برای خود ثبت کرده بود (شرکت .RSA Security Inc) اقامهٔ دعوی کرد که PGP با استفادهٔ بدون مجوز از الگوریتم RSA ، مرتکب نقض مالکیتِ حقوق معنوی آن شده است ولی از نسخهٔ ۲.۶ به بعد این مسئله رفع شد. همچنین PGP از الگوریتم IDEA که حق امتیاز آن هم ثبت شده، استفاده کرده است؛ این مسئله نیز مشکلاتی را ایجاد کرد.

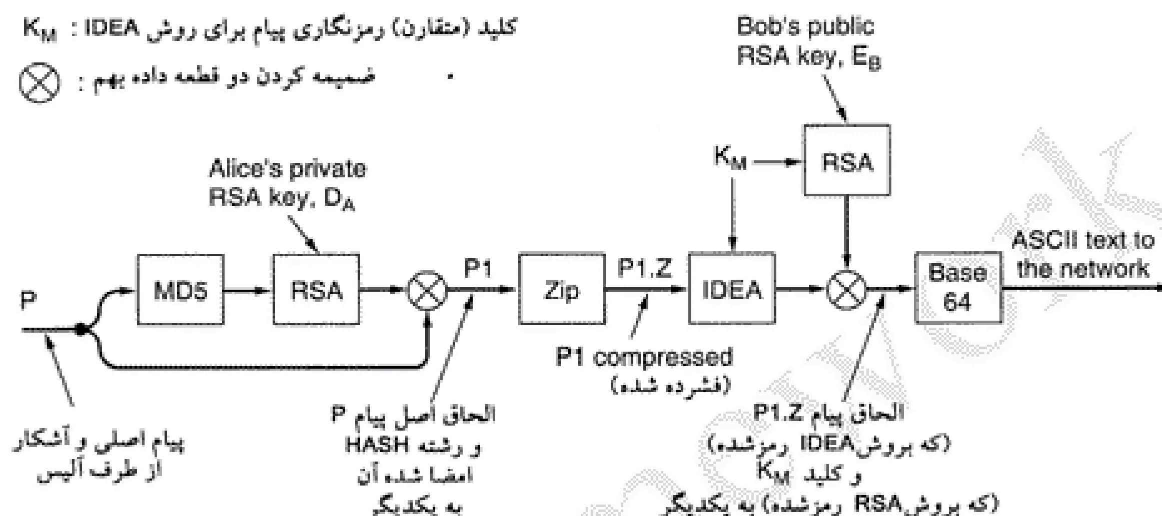
از آنجا که کدهای برنامهٔ PGP آزاد است لذا افراد و گروههای مختلف، نسخه‌های متفاوتی از آن را تولید و عرضه کردند. برخی از این نسخه‌ها با این هدف طراحی شده‌اند که قوانین «منع صدور تجهیزات حسّاس» را دور بزنند و برخی دیگر بر آن متمرکز بوده که از الگوریتمهای ثبت مالکیت شده استفاده نشود و حتی برخی دیگر خواستند آن را در قالب یک محصول تجاری با «کدهای بسته» و غیر آزاد عرضه کنند. اگرچه امروزه قانون صدور تجهیزات حساس ، نسبتاً آزادتر شده (البته به غیر از محصولات مبتنی بر AES که به هیچ وجه از ایالات متحده به خارج قابل صدور نیست) و در سپتامبر ۲۰۰۰ نیز حق مالکیت الگوریتم RSA منقضی گردید ولیکن میراث مشکلات و مسائل حقوقی PGP آن شد که نسخه‌های کاملاً ناسازگار از PGP با نامهای مختلف به جریان بیفتد. توضیح زیر بر روی نسخهٔ کلاسیک PGP که قدیمیترین و ساده‌ترین نسخهٔ آن است متمرکز خواهد بود. نسخهٔ رایج دیگر آن یعنی Open PGP در RFC 2440 تشریح شده است. همچنین نسخهٔ دیگری به نام GNU Privacy Guard نیز موجود است.

PGP تعمداً به جای ابداع یک روش رمزنگاری جدید از الگوریتمهای موجود استفاده کرده است. این روش، براساس الگوریتمهایی بنیان نهاده شده که تاکنون در مقابل بررسیها و حملات گسترده دوام آورده و شکسته نشده و همچنین آژانسهای دولتی در طراحی آنها تأثیرگذار نبوده‌اند. برای افرادی که به حکومت اعتماد ندارند این ویژگی، امتیاز بزرگی محسوب می‌شود.

PGP از فشرده‌سازی متن، سرّی‌سازی آن و امضاهای دیجیتالی حمایت می‌کند و امکانات بسیار جالبی در خصوص مدیریت کلید عرضه کرده است ولی امکانات کمی در خصوص تدوین و ارسال نامهٔ الکترونیکی دارد. PGP را می‌توان یک «**پیش پردازنده**» تلقی کرد که متن اصلی را به عنوان ورودی می‌گیرد و یک متن امضاء شده سرّی را که در قالب base64 کد شده است، باز می‌گرداند. البته خروجی آن بعداً می‌تواند توسط پست الکترونیکی

ارسال شود. برخی از پیاده‌سازیهای PGP در آخرین مرحله می‌توانند «عامل کاربر» [مثلاً نرم‌افزاری مثل Outlook] را جهت ارسال پیام فراخوانی کنند.

برای آن که ببینیم PGP چگونه کار می‌کند، اجازه بدهید مثال شکل ۸-۴۴ را بررسی کنیم. در این شکل آلیس می‌خواهد یک پیام آشکار امضاء شده مثل P را به روشی مطمئن برای باب بفرستد. باب و آلیس هر کدام دارای یک کلید خصوصی (D_X) و یک کلید عمومی (E_X) در الگوریتم RSA هستند. فعلاً فرض می‌کنیم هر کدام، کلید عمومی دیگری را می‌دانند؛ بعداً در خصوص مدیریت کلیدها به اختصار توضیح خواهیم داد.



شکل ۸-۴۴. عملکرد PGP برای ارسال یک پیام.

آلیس پشت کامپیوترش نشسته و کارش را با فراخوانی برنامه PGP آغاز می‌کند. PGP ابتدا به روش MD5 پیام P را درهم ریخته (Hash کرده) و یک رشتهٔ خلاصهٔ پیام (Message Digest) استخراج نموده و سپس آنرا طبق روش RSA با کلید خصوصی خودش D_A رمز می‌کند. بعداً وقتی باب پیام را دریافت کند می‌تواند رشتهٔ Hash را با کلید عمومی آلیس رمزگشایی کرده و صحّت آن را بررسی نماید. حتی اگر شخص ثالثی (مثل ترودی) بتواند رشته Hash را بدست آورده و آن را رمزگشایی کند، بدلیل استحکام روش MD5 ، این تضمین وجود دارد که نتواند پیام جعلی دیگری را با رشته Hash مشابه، تولید و جایگزین کند. بدین ترتیب در این مرحله صحّت و سلامت پیام (Integrity) تضمین می‌شود.

رشتهٔ Hash رمز شده و پیام اصلی به یکدیگر ضمیمه می‌شوند و یک پیام واحد به نام *P1* را ایجاد می‌کنند. سپس *P1* با استفاده از برنامهٔ ZIP که از الگوریتم «لیمپل-زیو» استفاده می‌کند فشرده می‌شود. (Lempel - Ziv, 1977). خروجی این مرحله را *P1.Z* بنامید.

PGP در ادامه، از آلیس یک ورودی تصادفی مطالبه می‌کند. برای تولید یک کلید ۱۲۸ بیتی بنام K_M برای الگوریتم رمزنگاری IDEA، از محتویات پیام و سرعت تایپ او (که هر دو تصادفی هستند) استفاده می‌شود. (K_M در ادبیات PGP کلید نشست نامیده می‌شود ولی در واقع کلید نشست، اسم بی‌مسمائی است چراکه در ارسال نامه هیچ نشستی ترتیب داده نمی‌شود). حال پیام *P1.Z* به روش IDEA و در حالت Feedback Mode[1]، با کلید K_M رمزنگاری می‌شود. مضاف براین، خود K_M نیز با استفاده از کلید عمومی باب یعنی E_B رمز می‌شود. نهایتاً این دو مولفه (یعنی کلید رمز+پیام فشرده و رمزشده) بهم ضمیمه شده و سپس در مبنای base64 که در بخش MIME از

۱. بخش ۸-۲-۳ را ببینید.

فصل هفتم تشریح کردیم، کدگذاری و تبدیل به متن ASCII می‌شود. پیام حاصل فقط شامل حروف الفباء ، ارقام و سمبولهای '+' ، '=' ، '/' خواهد بود بدین معنا که می‌تواند براحتی در بدنهٔ یک نامهٔ RFC 822 قرار گرفته و بدون هیچ تغییری به مقصد برسد.

وقتی باب پیام را دریافت می‌کند ابتدا عکس عمل کدگذاری base64 را انجام داده و کلید رمز IDEA را با کلید خصوصی RSA خود، استخراج می‌نماید. سپس با استفاده از این کلید، بدنهٔ پیام را رمزگشایی می‌کند تا *P1.Z* را بدست بیاورد. پس از آن که متن از حالت فشرده خارج شد، باب متن اصلی را از رشتهٔ رمزنگاری شدهٔ Hash جدا کرده و این رشته را رمزگشایی می‌کند. اگر رشتهٔ Hash متن اصلی با رشته MD5 ارسالی تطابق داشت او از صحت پیام P مطمئن شده و یقین حاصل می‌کند که پیام از طرف آلیس فرستاده شده است.

اشاره به این نکته ارزشمند است که در این ساختار، الگوریتم RSA فقط در دو جا استفاده شده است: (۱) برای رمزنگاری رشتهٔ ۱۲۸ بیتی MD5 (۲) برای رمزنگاری کلید ۱۲۸ بیتی IDEA. اگرچه الگوریتم RSA بسیار کُند است ولیکن فقط باید ۲۵۶ بیت را رمز کند نه حجم بسیار زیاد اطلاعات را! بعلاوه این ۲۵۶ بیت اطلاعات، مطلقاً تصادفی است و برای حدس کلید، ترودی به تلاش بسیار زیادی ، نیازمند خواهد بود. در عوض حجم سنگین عملیات رمزنگاری بدنهٔ پیام، به روش IDEA که بارها از RSA سریعتر است، انجام می‌شود. PGP، امنیت ، فشرده‌سازی و امضای دیجیتالی را بطور یکجا عرضه کرده و تمام این عملیات را بسیار سریعتر و مؤثرتر از ساختار پیشنهادی در شکل ۸-۱۹ ، انجام می‌دهد.

PGP از کلیدهای RSA با چهار اندازهٔ مختلف حمایت می‌کند. انتخاب طول مناسب بر عهدهٔ کاربر گذاشته شده است. این چهار انتخاب عبارتند از:

۱. **کلیدهای عِلّی (۳۸۴ بیت)**: امروزه می‌توان براحتی آن را شکست.
۲. **کلیدهای تجاری (۵۱۲ بیت)**: فقط توسط مؤسسات سه حرفی قابل شکستن است.[۱]
۳. **کلیدهای نظامی (۱۰۲۴ بیتی)**: هیچکس بر روی کره زمین نمی‌تواند آن را بشکند.
۴. **کلیدهای کیهانی (۲۰۴۸ بیتی)**: هیچکس حتی در سیارات دیگر نمی‌تواند آن را بشکند!!

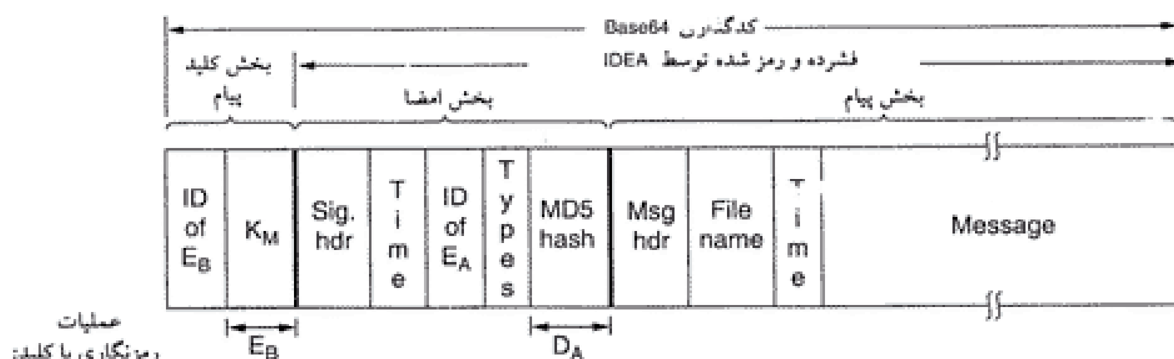
از آنجایی که RSA فقط در دو محاسبهٔ کوچک [یعنی برای رمزنگاری ۲ کلید ۱۲۸ بیتی] بکار گرفته می‌شود منطقی است که عموم افراد از کلید با طول ۲۰۴۸ بیت استفاده کنند.

قالب یک پیام کلاسیک PGP در شکل ۸-۴۵ نشان داده شده است. البته از قالبهای بیشمار دیگری نیز استفاده می‌شود. پیام در سه بخش شامل بخش کلید IDEA، بخش امضاء و بخش بدنهٔ پیام سازماندهی شده است. بخش حاوی کلید نه تنها کلید را در بر می‌گیرد بلکه شمارهٔ شناسایی کلید عمومی را نیز شامل می‌شود. بدین ترتیب کاربران مجاز به داشتن چندین کلید عمومی هستند.

بخش امضاء شامل یک سرآیند خاص است که در اینجا به آن نخواهیم پرداخت. پس از سرآیند امضاء، مهر زمان و سپس شمارهٔ شناسایی کلید عمومی فرستنده درج می‌شود که توسط این کلید عمومی، رشتهٔ Hash امضاء دیجیتالی از رمز خارج خواهد شد. سپس در ادامه، یک فیلد Type قرار داده شده که نوع الگوریتم بکار رفته را تعیین می‌کند (تا مثلاً اجازه بدهد در صورت ابداع MD6 یا RSA2 از آنها استفاده شود!) سپس در انتهای بخش امضاء، رشتهٔ کد MD5 پیام قرار می‌گیرد.

بخش پیام نیز دارای یک فیلد سرآیند است، سپس نام پیش فرض فایل درج می‌شود تا اگر گیرندهٔ پیام خواست آنرا بر روی دیسک ذخیره کند از این نام استفاده شود. سپس مهر زمانِ ایجادِ پیام، درج شده و نهایتاً پیام اصلی قرار

۱. منظور از مؤسسات سه حرفی NSA، FBI، CIA و امثال آنهاست! -م.



شکل ۸-۴۵. قالب پیامهای PGP.

می‌گیرد.

در PGP به مدیریت کلیدها توجه ویژه‌ای شده است چرا که پاشنهٔ آشیل تمام سیستمهای امنیتی محسوب می‌شود! مدیریت کلیدها در PGP بدین نحو انجام می‌گیرد که هر کاربر به صورت محلی از دو ساختمان دادهٔ خاص نگهداری می‌کند: الف) «دسته کلید خصوصی» ب) «دسته کلید عمومی». «**دستهٔ کلید خصوصی**» شامل یک یا چند زوج کلید عمومی و همتای خصوصی آنهاست. دلیل آن که از چندین زوج کلید برای هر کاربر حمایت می‌شود آنست که کاربر بتواند بطور متناوب کلید عمومی خود را عوض کند، یا هر گاه شخصی شکّ کرد که یکی از کلیدهای خصوصیش فاش شده بلافاصله آن را تغییر بدهد. هر زوج کلید دارای یک شناسه (شمارهٔ شناسایی) متناظر است که براساس آن فرستندهٔ پیام به گیرنده اطلاع می‌دهد که برای رمزگشایی پیام از کدام کلید باید استفاده کند. شناسهٔ کلیدها، ۶۴ بیتِ کم‌ارزش هر کلید عمومی است. کاربران خودشان وظیفه دارند که از انتخاب کلیدهایی که ۶۴ بیت کم ارزششان یکسان است خودداری کنند. کلیدهای خصوصی قبل از ذخیره بر روی دیسک با استفاده از یک کلمهٔ عبور (با اندازهٔ طولانی) رمز می‌شوند تا از خطر سرقت مصون بمانند.

«**دسته کلید عمومی**» در برگیرندهٔ کلیدهای عمومی متعلق به کاربران طرف مقابل ارتباط است. برای رمزنگاری کلیدهای هر پیام، به کلید عمومی گیرندهٔ پیام نیز نیاز است. هر درایه (Entry) در دستهٔ کلیدهای عمومی، نه تنها شامل خود کلید است بلکه شناسهٔ ۶۴بیتی هر کلید عمومی (۶۴بیت کم ارزش کلید) و فیلد دیگری که در آن میزان اعتماد کاربر به آن کلید مشخص شده است را نیز در برمی‌گیرد. در زیر فلسفهٔ این فیلد مشخص شده است.

مسئله‌ای که ممکن است برای PGP کلاسیک بوجود بیاید آنست که فرض کنید کلیدهای عمومی بر روی «بولتن برد» (Bulletin Board) اعلام شده باشد. یک راه برای آنکه ترودی بتواند نامه‌های محرمانهٔ باب را بخواند آن است که به این سایت حمله کرده و کلید عمومی او را به دلخواه خود تغییر بدهد. بعداً وقتی آلیس این کلید را به خیال آن که به باب تعلق دارد دریافت می‌کند، ترودی خواهد توانست در میانهٔ راه نامه‌های او را استراق سمع نماید.[1]

برای پیشگیری از این گونه حملات یا حداقل کاهش تبعات آن، آلیس نیازمند آن است که بداند به چه اندازه می‌تواند به آیتمی که در دستهٔ کلیدهای عمومی، به باب منسوب شده اعتماد کند. فیلد «**میزان اعتماد**» به همین منظور بکار می‌رود. اگر آلیس کلید عمومی باب را بر روی یک فلاپی دیسک شخصاً از باب گرفته باشد، می‌تواند فیلد «**میزان اعتماد**» این کلید را به مقدار بالایی تنظیم نماید. این روش مدیریت کلید که بصورت غیرمتمرکز و تحت نظارت کاربر انجام می‌شود، PGP را از قید وابستگی به یک PKI متمرکز رها کرده است.

علیرغم این، افراد معمولاً کلید عمومی دیگران را با سؤال از یک سرویس دهندهٔ مورداعتماد توزیع کلید،

۱. حملهٔ «گروه آتش‌نشان» (bucket brigade) را در بخش ۸-۷-۲ مطالعه نمائید.

بدست می‌آورند. به همین دلیل پس از آن که X.509 استاندارد شد، PGP در کنار مکانیزم سنتی خود از گواهینامهٔ دیجیتالی X.509 نیز حمایت کرد. تمام نسخه‌های فعلی PGP از گواهینامه‌های X.509 پشتیبانی می‌کنند.

۸-۸-۲ PEM (Privacy Enhanced Mail)

برخلاف PGP که از ابتدا یک نمایش تکنفره با هنرنمایی زیمرمن بود، دومین مثال ما، PEM ، یک استاندارد رسمی اینترنت است که در اواخر دهۀ هشتاد توسعه یافت و در چهار RFC از RFC 1421 تا RFC 1424 تشریح شد. اصول PEM تقریباً با PGP مشابه است: هر دو به منظور احراز هویت و محرمانه نگاه داشتن نامه‌ها در سیستمهای پست الکترونیکی مبتنی بر RFC 822 هستند؛ ولیکن در روشها و فناوری بکار رفته در آن با PGP تفاوتهایی دارد.

پیامهای ارسالی توسط PEM در ابتدا به قالب استاندارد تبدیل می‌شود به نحوی که تمام نامه‌ها از قواعد و استاندارد مشابهی در خصوص فضاهای خالی (همانند فاصله‌ها، Tab و...) پیروی کنند. سپس با استفاده از MD2 یا MD5 برای نامه یک رشته Hash بدست می‌آید. پس از الحاق رشته Hash به پیام، مجموع این دو با استفاده از روش DES رمزنگاری می‌شود. با در نظر داشتن ضعف کلیدهای ۵۶بیتی در DES، این انتخاب کاملاً مشکوک به نظر می‌رسد. سپس پیام رمز شده به روش base64 کُدگذاری و برای گیرندۀ آن ارسال می‌شود.

همانند PGP ، هر پیام با استفاده از کلید یک‌مرحله‌ای و متقارن رمز شده و آن کلید نیز در کنار پیام جاسازی و ارسال می‌شود. این کلید با استفاده از روش رمزنگاری RSA یا روش Triple DES (DES سه گانه مبتنی بر مکانیزم EDE) رمز می‌گردد.

مدیریت کلید در PEM نسبت به PGP سازمان یافته‌تر است. سیدها باگواهینامه‌های X.509 که توسط مراکز صدور گواهی -CA- صادر شده‌اند، تصدیق می‌شوند. مراکز صدور گواهی در یک ساختار سلسله مراتبی که از مرکزی به نام «ریشه» (Root) شروع می‌شود، سازماندهی شده‌اند. حسن این ساختار آنست که ابطال گواهینامه‌ها با انتشار متناوب فهرست CRL[1] توسط «ریشه»، بسادگی میسر می‌باشد.

تنها مشکل PEM آن است که هیچکس تاکنون از آن استفاده نکرده و تقریباً به تاریخ پیوسته است! معضل آن بیشتر سیاسی بود: چه کسی و تحت چه شرایطی باید مسئولیت «ریشه» (Root) را بر عهده بگیرد؟ نامزدهای این مسئولیت کم نبودند ولی بسیاری افراد از اعتماد به یک شرکت واحد می‌ترسند. جدی‌ترین نامزد این مسئولیت، شرکت RSA بود که می‌خواست به ازای صدور هر گواهینامه مبلغی را دریافت کند. بعضی مؤسسات از این ایده استقبال نکردند. مخصوصاً از این جهت که دولت ایالات متحده مجاز به استفاده از تمام ابداعات و اختراعات ثبت شده در آن کشور است و همچنین شرکتهای خارج از ایالات متحده عادت داشتند از الگوریتم RSA به رایگان استفاده کنند (شرکت RSA فراموش کرده بود امتیاز استفاده از RSA را در خارج از ایالات متحده به نفع خود ثبت کند) مؤسسات مختلف علاقمند نبودند برای کاری که قبلاً به رایگان انجام می‌شد به شرکت RSA پول بپردازند. نهایتاً هیچکس برای پذیرش مسئولیت «ریشه» [در ساختار سلسله مراتبی صدور گواهینامه] پیدا نشد و PEM شکست خورد.

۸-۸-۳ S/MIME

اقدام بعدی IETF برای امنیت سیستم پست الکترونیکی که S/MIME نامیده شد در اسناد RFC 2632 تا RFC 2643 تشریح شده است. همانند PEM ، این روش نیز امکان «احراز هویت»، «تایید صحت اطلاعات»، «سرّی نگاه داشتن پیام» و «غیرقابل انکار بودن پیام» را تضمین کرده است. همچنین این روش کاملاً قابل انعطاف

---

۱. CRL: «فهرست گواهینامه‌های باطل شده»

بوده و از الگوریتمهای رمزنگاری مختلفی پشتیبانی می‌کند. همانگونه که از نام S/MIME مشخص است این روش، استانداردِ MIME را تکمیل و امن کرده است فلذا عجیب نیست که از انواع مختلف پیامها حمایت و حفاظت می‌کند.[1] در این استاندارد تعدادی سرآیند جدید برای MIME تعریف شده که برای عملیاتی نظیر مشخص کردن امضای دیجیتالی پیام، کاربرد دارند.

IETF قطعاً از تجربهٔ PEM چیزهایی را آموخته بود. S/MIME به یک ساختار سلسله‌مراتبی صدور گواهینامه که از «ریشه» شروع شود نیاز ندارد. کاربران می‌توانند به جای یک مؤسسهٔ واحد به عنوان ریشه، مجموعه‌ای از مؤسسات مورد اعتمادِ صدور گواهینامه (Trust Anchors) را به خدمت بگیرند. هر گاه در سلسله‌مراتب تائید گواهینامه‌های دیجیتالی، گواهینامه‌ای به یکی از این مؤسسات مورداعتماد کاربر ختم گردد، معتبر فرض می‌شود. S/MIME از پروتکلها و استانداردهایی استفاده می‌کند که تا اینجا همهٔ آنها را بررسی کرده‌ایم و بیش از این، بدانها نخواهیم پرداخت. برای دسترسی به شرح مبسوط آن به RFCهای مربوطه مراجعه نمائید.

۹-۸ امنیت وب

تا اینجا دو زمینهٔ بسیار مهم را که مقولهٔ امنیت در آنها ضروری است، مطالعه کرده‌ایم: امنیت در مخابرهٔ داده‌ها و امنیت در سیستم پست الکترونیکی. می‌توانید اینها را به عنوان سوپ و پیش‌غذا تلقی کنید. حال وقت آن رسیده تا به موضوع اصلی بپردازیم: «**امنیت وب**». امروزه وب جولانگاه اخلالگران و ترودیهایی است که به کارهای کثیف خود مشغولند. در بخشهای آتی به برخی از موارد و مشکلات مربوط به امنیت وب نگاهی خواهیم انداخت.

بطور تقریبی امنیت وب را می‌توان در سه بخش تقسیم کرد: **اول** آن که چگونه اشیاء (Objects) و منابع (Resources) به روشی مطمئن نامگذاری شوند؟ **دوم** آنکه چگونه می‌توان ارتباطی تایید شده و مطمئن برقرار کرد؟ **سوم** وقتی یک وب‌سایت برای مشتری خود یک قطعه کُد قابل اجرا می‌فرستد چه اتفاقی می‌افتد؟ پس از آنکه به تهدیدهای بالقوه در وب نگاهی انداختیم این سه مورد را نیز بررسی خواهیم کرد.

۸-۹-۱ تهدیدها

تقریباً هر هفته در روزنامه‌ها و مجلات، مطالبی در خصوص مشکلات امنیتی سایتهای وب می‌خوانید. این وضعیت واقعاً فاجعه است. بیائید مثالهایی از آنچه که در گذشته اتفاق افتاده را بررسی کنیم. اولاً صفحهٔ وب اصلی (Home Page) بسیاری از مؤسسات مورد حمله قرار گرفته و با صفحات وب دلخواه اخلالگران (Crackers) تعویض شده است. (مطبوعات عموماً به کسی که حریم کامپیوترها را درهم می‌شکند، **نفوذگر** یا *Hacker* می‌گویند در حالی که بسیاری از برنامه‌نویسان این اصطلاح را برای یک برنامه‌نویس نخبه بکار می‌برند. ما نیز ترجیح می‌دهیم این افراد را اخلالگر یا «Cracker» بنامیم.) سایتهایی که تاکنون درهم شکسته و در آنها اخلال شده شامل سایتهای مشهوری مثل Yahoo ، U.S.Army ، CIA ، NASA و New York Times بوده است. در بسیاری موارد، اخلالگران پس از حمله، جملات و متون مسخره در این سایتها درج کرده‌اند و پس از چند ساعت این سایتها اصلاح شده‌اند.

حال موارد خطرناکتر را بررسی می‌نمائیم. تعداد بی‌شماری از سایتها توسط حملهٔ DoS (حملهٔ نوع اخلال در سرویس‌دهی) از کار افتاده‌اند و اخلالگر موفق شده یک سایت را با ترافیکی سیل‌آسا مواجه کند به نحوی که قادر به پاسخ‌دهی به تقاضاهای مجاز نباشد. گاهی اوقات این حمله توسط تعداد بسیار زیادی از ماشینها بر علیه یک

---

[1] در فصل قبل دانستید که MIME تقریباً از هر نوع پیامی (با هر قالب و محتوا) حمایت می‌کند. طبعاً S/MIME نیز باید همینگونه باشد. —م.

سایت شکل گرفته که این ماشینها خود قربانی حملات قبلی اخلالگران بوده‌اند[۱] (حملهٔ نوع DDoS[۲]). این نوع حمله به قدری رایج شده که برای هیچکس خبر جدیدی تلقی نمی‌شود ولی سایت مورد حمله گاهی هزاران دلار متضرر می‌گردد.

در سال ۱۹۹۹ یک اخلالگر سوئدی در سایت وب Microsoft Hotmail رسوخ کرد و با ایجاد یک سایت معادل (Mirror Site) که در آن کاربران نام کاربری و کلمهٔ عبور خود را درج می‌کردند موفق شد نامه‌های جاری و بایگانی شده تمام افراد مراجعه کننده به این سایت را بخواند.

در یک مورد دیگر، اخلالگر ۱۹ سالهٔ روسی که ماکسیم نام داشت توانست به یک سایت تجارت الکترونیکی نفوذ کند و شمارهٔ ۳۰۰۰۰۰ کارت اعتباری را بدزدد. او نزد صاحب سایت رفت و به او گفت که اگر صد هزار دلار به او پرداخت نکنند تمام این شماره‌ها را بر روی اینترنت منتشر خواهد کرد. آنها به خواستهٔ او اعتنایی نکردند و او نیز با لجاجت شمارهٔ کارتهای اعتباری دیگران را بر روی اینترنت گذاشت و ضرر و زیان زیادی به شماری از قربانیان بی‌گناه تحمیل کرد.

در یک حملهٔ دیگر، یک دانشجوی ۲۳ ساله کالیفرنیایی گزارشی را برای یک آژانس خبری ارسال کرد و به دروغ عنوان نمود که شرکت «امیولکس» (Emulex Corporation) می‌خواهد در تراز اقتصادی سه ماههٔ خود، حجم زیان بسیار بالایی را اعلام کند و مسئولان آن بلافاصله استعفا خواهند کرد. در عرض چند ساعت، سهام این شرکت ۶۰ درصد سقوط کرد و سهامداران بیش از دو میلیارد دلار متضرر شدند!! عوامل این تخلف با معاملهٔ سهام خود قبل از ارسال این گزارش، حدود یک چهارم میلیون دلار بدست آوردند. اگرچه این حادثه در اثر حمله و نفوذ به یک وب‌سایت نبود ولیکن روشن است که قرار دادن چنین گزارشهایی در صفحهٔ اصلی یک شرکت بزرگ، تأثیر مشابه و وخیمی بجا خواهد گذاشت.

متأسفانه می‌توانیم از اینگونه حملات، صفحات بسیار زیادی مطلب و گزارش بنویسیم ولیکن زمان آن فرا رسیده که به موارد فنی در خصوص امنیت وب بپردازیم. برای اطلاعات بیشتر در خصوص انواع مختلف و مستند مشکلات امنیتی وب به مراجع (Anderson, 2001; Garfinkel with Spafford, 2002; Schneier, 2000) مراجعه نمائید. با جستجو در اینترنت نیز تعداد بی‌شماری از موارد مشابه را خواهید یافت.

۸-۹-۲ نامگذاری مطمئن

اجازه بدهید با یک مسئله بسیار بنیادی شروع کنیم: آلیس تمایل دارد وب‌سایت متعلق به باب را ببیند. لذا آدرس URL باب را در نرم‌افزار مرورگر خود درج کرده و چند ثانیه بعد یک صفحهٔ وب بر روی نمایشگر او ظاهر می‌شود. آیا این صفحه واقعاً متعلق به باب است؟ ممکن است باشد و احتمال دارد نباشد! ممکن است ترودی مجدداً از حقه قدیمی خود استفاده کرده باشد. به عنوان مثال ممکن است او در میانهٔ راه تمام بسته‌های خروجی آلیس را دریافت و بازرسی کند. وقتی ترودی در بین بسته‌ها به یک تقاضای HTTP GET که به سمت وب‌سایت باب روانه شده بر می‌خورد، می‌تواند خود به وب‌سایت باب مراجعه کرده و آن صفحه را دریافت و تغییرات مورد نظر خود را در آن ایجاد کند و صفحهٔ جعلی را به آلیس برگرداند. حال ترودی می‌تواند (مثلاً) قیمت کالاهای فروشگاه الکترونیکی باب را به قدری پایین آورد تا جلب توجه کرده و آلیس را فریب بدهد تا شمارهٔ کارت اعتباری خود را جهت خرید کالا از «باب» ارسال نماید.

---

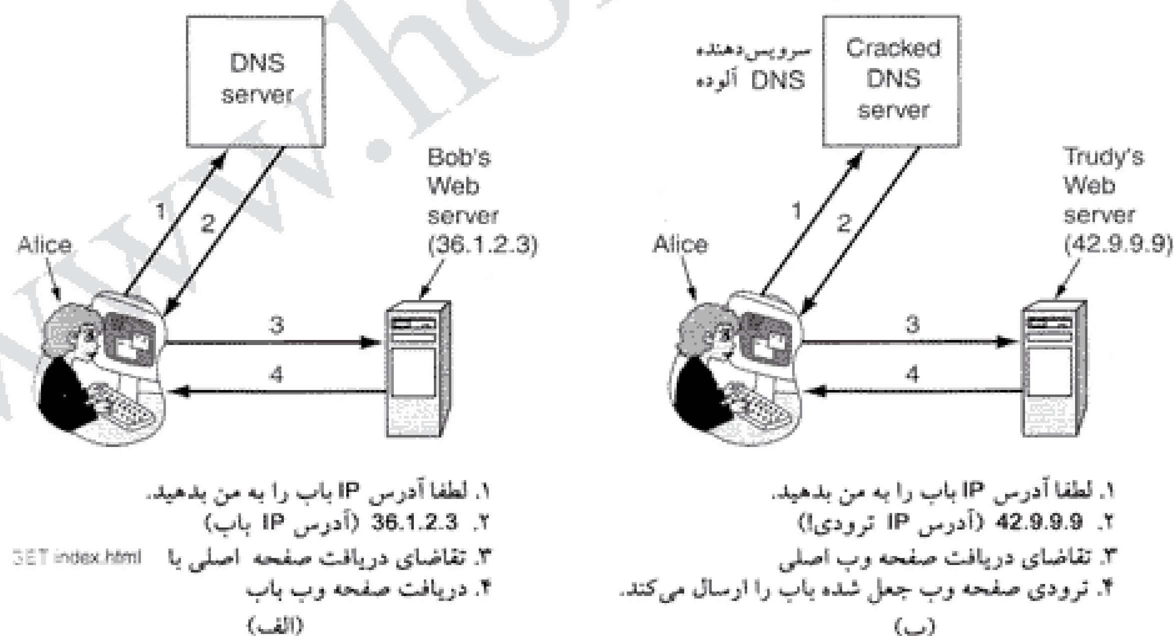
۱. به ماشینهایی که خود قربانی نفوذ بدخواهان شده‌اند و ناخودآگاه در حمله به یک سایت شرکت می‌کنند اصطلاحاً «ماشینهای زامبی» (Zambie) گفته می‌شود. -م.

۲. Distributed Denial of Service

یکی از اشکالات حملهٔ mitm[1] (از دیدگاه اخلالگران) آنست که ترودی مجبور خواهد بود در موقعیتی قرار بگیرد تا بتواند ترافیک خروجی آلیس را دریافت کرده و ترافیک ورودی آن را دستکاری و جعل نماید. در عمل او باید از خط تلفن آلیس یا باب انشعاب بگیرد چرا که انشعاب از ستون فقرات شبکه‌ای که با فیبرنوری ایجاد شده بسیار دشوار و ناموفق است. اگرچه ایجاد انشعاب فعال از خطوط تلفن قطعاً ممکن است ولیکن باید کارهای فیزیکی دشواری انجام بشود و چون ترودی گذشته از آنکه زیرک است، تنبل هم هست برای فریب دادن آلیس، راه ساده‌تری را در پیش می‌گیرد:

DNS Spoofing

به عنوان نمونه فرض کنید ترودی قادر به نفوذ در سیستم DNS شده است و توانسته در حافظهٔ نهان DNS (DNS Cache) متعلق به ISP آلیس، تغییر ایجاد کرده و آدرس IP باب (مثلاً 36.1.2.3) را با آدرس IP خودش (42.9.9.9) عوض کند. این دستکاری و تغییر، حملهٔ ذیل را در پی خواهد داشت. برای شروع، عملکرد معمول سیستم DNS را در شکل ۸-۴۶-الف در نظر بگیرید. در این شکل: (۱) آلیس از DNS، آدرس IP باب را سؤال می‌کند. (۲) پاسخ آن را دریافت می‌کند. (۳) از باب، صفحهٔ اصلی سایت (Home Page) او را تقاضا می‌کند. (۴) صفحه را دریافت می‌نماید. پس از آن که ترودی آدرس IP رکورد DNS باب را با آدرس IP خودش عوض کرد، وضعیت ۸-۴۶-ب را خواهیم داشت. در اینجا وقتی آلیس آدرس IP باب را درخواست می‌کند، آدرس ترودی را تحویل خواهد گرفت لذا کل ترافیک داده‌هایی که باید به سمت باب هدایت شود به سوی ترودی می‌رود. حال ترودی می‌تواند حملهٔ mitm را پایه‌ریزی کند بدون آن که نیازی به انشعاب گرفتن از خط تلفن باب یا آلیس داشته باشد. او فقط باید بتواند در سرویس دهندهٔ DNS نفوذ کرده و تنها یک رکورد را تغییر بدهد که بسیار ساده‌تر از انشعاب گرفتن از خطوط ارتباطی است!



شکل ۸-۴۶. (الف) وضعیت طبیعی. (ب) حمله‌ای بر اساس دستکاری در داده‌های DNS و تغییر در رکورد مربوط به باب.

[1] Man In the Middle

ترودی چگونه می‌تواند DNS را فریب داده و دستکاری کند؟ این کار نسبتاً ساده به نظر می‌رسد! بطور اجمالی، ترودی می‌تواند سرویس دهندهٔ DNS در ISP آلیس را وادار به ارسال تقاضائی جهت ترجمهٔ آدرس باب نماید. متأسفانه چون DNS از UDP بهره می‌گیرد، سرویس دهندهٔ DNS هیچ راهی برای بررسی آن که واقعاً چه کسی پاسخ این تقاضا را داده، ندارد. ترودی می‌تواند از این ویژگی سوءاستفاده کرده و پاسخ مورد نظر DNS را جعل و بدین نحو آدرس IP غلطی را در حافظهٔ نهان سرویس‌دهندهٔ DNS (DNS Cache) تزریق نماید. برای ساده‌تر شدن بحث فرض می‌کنیم که سرویس دهندهٔ DNS در ISP آلیس، هیچ درایه‌ای (Entry) در خصوص آدرس وب‌سایت باب ، *bob.com* ، ندارد. البته اگر چنین درایه‌ای وجود داشته باشد ترودی می‌تواند آنقدر صبر کند تا زمان اعتبار آن منقضی شود و بعداً تلاش خود را از سر بگیرد. (یا از روشهای دیگر استفاده کند.)

ترودی حملهٔ خود را با ارسال تقاضای ترجمهٔ آدرس bob.com به سمت سرویس دهندهٔ DNS در ISP آلیس، آغاز می‌نماید. از آنجایی که هیچ درایه‌ای برای این نام در حافظهٔ DNS وجود ندارد، سرویس‌دهندهٔ DNS از سرویس دهندهٔ سطح بالا یعنی سرویس دهندهٔ com. در خصوص این نام سؤال می‌کند. حال ترودی در نقش سرویس دهندهٔ DNS سطح بالا حمله کرده و یک پاسخ جعلی و دروغین با این مضمون برای DNS می‌فرستد: **«نام *bob.com* معادل با 42.9.9.9 است»!!** در حالی که این آدرس IP متعلق به خود اوست. اگر پاسخ جعلی و غلط او زودتر به ISP آلیس برسد در حافظهٔ سرویس دهندهٔ DNS درج می‌شود و پاسخ واقعی که بعداً می‌رسد به عنوان پاسخی بیجا و سرزده کنار گذاشته خواهد شد. فریب دادن سرویس دهندهٔ DNS به نحوی که آدرس IP جعلی و دروغین را در حافظهٔ نهان خود درج کند اصطلاحاً «DNS Spoofing» نامیده می‌شود.[1] به حافظهٔ نهان که در آن آدرسهای IP غلط درج شده است اصطلاحاً **«حافظهٔ نهان سمّی»** (Poisoned Cache) گفته می‌شود.

البته انجام این کارها با دشواریهایی روبروست: اول آن که ISP آلیس، بررسی می‌کند تا ببیند آیا بستهٔ پاسخ دریافتی، آدرس IP سرویس دهندهٔ معتبر و واقعی سطح بالا را در خود حمل می‌کند یا خیر؟ از آنجایی که ترودی می‌تواند هر چه که خواست در فیلد آدرس IP بستهٔ پاسخ قرار بدهد لذا قادر است این بررسی را ناکام بگذارد زیرا آدرسهای IP سرویس دهنده‌های سطح بالا شناخته شده هستند و او می‌تواند از این آدرسها استفاده کرده و پاسخ جعلی خود را با این آدرسها بفرستد.

دوم آنکه سرویس دهندهٔ DNS برای تشخیص اینکه کدام پاسخ متعلق به کدام تقاضا بوده است برای هر بستهٔ تقاضا یک شمارهٔ ترتیب در نظر می‌گیرد. برای آن که ISP آلیس فریب داده شود، ترودی مجبور است شمارهٔ ترتیب تقاضای جاری را بداند. ساده‌ترین راه برای فهمیدن شمارهٔ تقاضای جاری آنست که ترودی برای خودش یک نام حوزه مثل *trudy-the-intruder.com* ثبت کند. فرض کنیم آدرس IP او 42.9.9.9 است. ترودی همچنین یک سرویس دهندهٔ DNS برای آدرس ثبت شده خودش به نام *dns.trudy-the-intruder.com* ایجاد می‌کند. البته آدرس IP این سرویس دهنده نیز 42.9.9.9 است چراکه ترودی یک ماشین بیشتر در اختیار ندارد. حال باید ISP آلیس را از وجود DNS خودش مطلع نماید. این کار بسیار ساده است؛ تمام کاری که باید انجام شود آنست که از ISP آلیس، در مورد نامی مثل *foobar.trudy-the-intruder.com* سؤال نماید. این کار ISP آلیس را وادار می‌کند تا از سرویس‌دهندهٔ سطح بالای *com*. در خصوص نام حوزهٔ جدید ترودی سؤال کند.

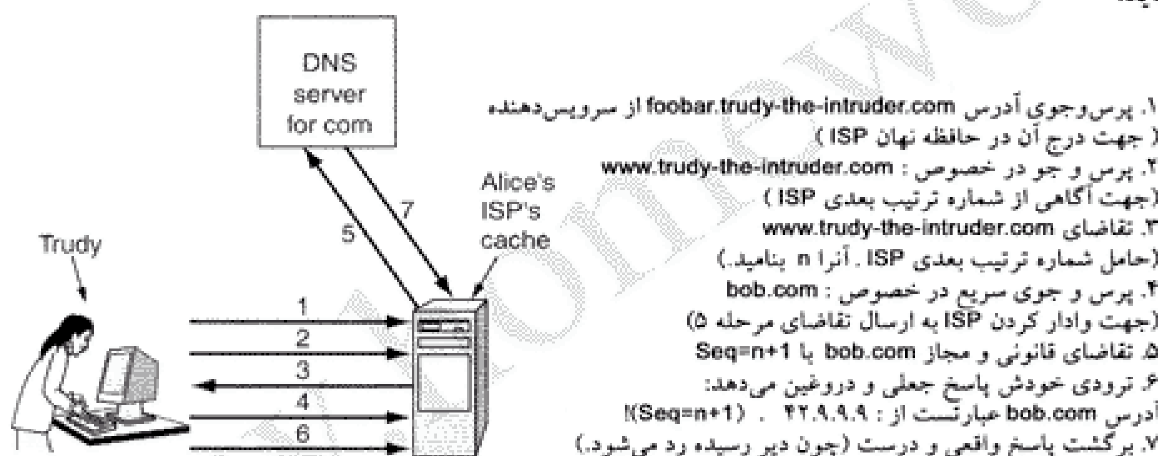
پس از آنکه آدرس *dns.trudy-the-intruder.com* در حافظهٔ نهان DNS آلیس درج شد حملهٔ اصلی می‌تواند شروع شود. ترودی تقاضایی به سمت DNS متعلق به ISP آلیس می‌فرستد و ترجمهٔ نام *www.trudy-the-intruder.com* را خواستار می‌شود. این ISP طبق معمول یک بستهٔ پرسش به سمت سرویس

---

[1] یعنی DNS متعلق به ISP آلیس را وادار به پرسش در خصوص نام bob.com می‌کند و بلافاصله خودش به این پرسش جواب غلط می‌دهد؛ چون DNS از UDP بهره می‌گیرد تشخیص هویت پاسخ‌دهنده میسر نیست. -م.

دهندهٔ ترودی ارسال می‌کند. این بستهٔ تقاضا شمارهٔ ترتیب مورد نیاز ترودی را با خود حمل می‌نماید. ترودی بلافاصله از ISP آلیس در مورد نام حوزهٔ باب (bob.com) سؤال می‌کند. سپس به سرعت به سؤال خودش پاسخ داده و بسته‌های جعلی را که در ظاهر به نظر می‌رسد از سرویس‌دهنده‌های سطح بالا رسیده برای آن ISP می‌فرستد. شمارهٔ ترتیب این بسته‌های جعلی یکی بیشتر از بسته‌ای است که چند لحظه قبل دریافت شده بود. البته برای اطمینان بیشتر ترودی می‌تواند یک بسته پاسخ با شماره ترتیب دو واحد بیشتر یا دهها بسته با شمارهٔ ترتیب متوالی (با شمارهٔ بیشتر) بفرستد. بالاخره یکی از آنها با شمارهٔ ترتیب تقاضای ارسالی منطبق خواهد شد و بقیه حذف خواهند گردید. وقتی پاسخ جعلی توسط DNS آلیس دریافت شد در حافظهٔ نهان ذخیره می‌شود. هرگاه در آینده پاسخ واقعی دریافت شود با توجه به آن که قبلاً پاسخ جعلی دریافت شده، کنار گذاشته خواهد شد.

حال وقتی آلیس آدرس bob.com را جستجو می‌کند به او گفته می‌شود که از آدرس 42.9.9.9 استفاده کند که همان آدرس ترودی است. بدین ترتیب ترودی موفق شده از اتاق محل زندگی خود، حملهٔ mitm را برعلیه آلیس پی‌ریزی و اجرا نماید. گامهای بعدی این حمله در شکل ۸–۴۷ نشان داده شده است. بدتر از همه، روش فوق تنها راه فریب دادن DNS نیست. راههای دیگری هم هستند که ترودی می‌تواند در صورت عدم موفقیت آنها را نیز بیازماید.



شکل ۸–۴۷. چگونگی فریب دادن ISP آلیس توسط ترودی .

Secure DNS

این حملهٔ خاص را می‌توان بدین نحو خنثی کرد که DNS بجای استفاده از شمارهٔ ترتیب شمارشی برای بسته‌های پرسش و پاسخ از شماره‌های کاملاً تصادفی بهره بگیرد ولی به نظر می‌رسد که هر گاه یک رخنهٔ نفوذ مسدود شود، در کنار آن رخنهٔ دیگری در سیستم ایجاد می‌شود. مشکل اصلی آن است که DNS در زمانی طراحی شده که اینترنت فقط یک ابزار تحقیقاتی و آن هم در چند صد دانشگاه بیشتر نبود و هیچیک از افراد نظیر آلیس ، باب یا ترودی به این محفل دعوت نشده بودند. در آن زمان امنیت ، مورد مهمی به حساب نمی‌آمد؛ مهمترین مورد آن بود که اینترنت در همه حال کار کند. در طول این سالها شرایط حاکم بر محیط اینترنت بشدت تغییر کرد؛ بهمین دلیل در سال ۱۹۹۴، IETF گروهی را مأمور ساخت تا DNS را بطور بنیادی امن نمایند. این پروژه به نام DNSsec شناخته می‌شود و نتیجهٔ آن در RFC 2535 ارائه شده است. متأسفانه از DNSsec در ترکیب شبکه‌ها به صورت گسترده و کامل استفاده نمی‌شود و به همین دلیل هنوز هم بسیاری از سرویس دهنده‌های DNS نسبت به حملهٔ DNS Spoofing آسیب‌پذیر هستند.

DNSsec ، از دیدگاه مفهومی فوق‌العاده ساده است. این سرویس دهنده بر اصول رمزنگاری با کلید عمومی

استوار است. هر ناحیه در DNS (DNS Zone) (با دیدگاه شکل ۷-۴) دارای یک زوج کلید عمومی و خصوصی است. تمام اطلاعاتی که توسط سرویس دهندهٔ DNS فرستاده می‌شود قبل از ارسال، با کلید خصوصی **«ناحیهٔ مبدأ»**[1] امضاء (رمز) شده و بدین ترتیب گیرندهٔ آنها می‌تواند هویت این اطلاعات را بررسی نماید.

DNSsec سه دسته خدمات اساسی زیر را ارائه می‌کند:

۱. اثبات آن که داده‌ها از کجا منشاء گرفته‌اند و به چه کسی متعلقند.
۲. توزیع کلید عمومی
۳. احراز هویت تقاضاها و تعاملاتی که با سرویس‌دهنده صورت می‌گیرد.

اصلی‌ترین سرویس همان مورد اول است که براساس آن بررسی می‌شود که اطلاعات برگشتی از DNS واقعاً متعلق به چه کسی است. مورد دوم برای ذخیره و بازیابی مطمئن کلیدهای عمومی کاربرد دارد. سومین مورد بدان جهت نیاز است تا بتوان حملهٔ نوع تکرار (Replay Attack) و حمله DNS Spoofing را خنثی کرد. دقت کنید که در DNSsec خدمات رمزنگاری داده‌ها ارائه نمی‌شود زیرا تمام اطلاعات DNS، عمومی و غیرمحرمانه است. انتظار می‌رود مراحل جایگزینی DNSsec با DNS معمولی چندین سال طول بکشد لذا سرویس دهندهٔ DNSsec باید این قابلیت اساسی را داشته باشد که با سرویس‌دهنده‌های معمولی و ناامن نیز کار کند و طبعاً پروتکل DNSsec در مقایسه با DNS ، تغییر بنیادی و ناسازگار نداشته است. اجازه بدهید نگاهی به این پروتکل بیندازیم.

رکوردهای DNS در گروههایی به نام RRSets (Resource Record Sets) دسته‌بندی شده‌اند. تمام رکوردهایی که «نام دامنه» ، «نوع» و «کلاس» مشابه دارند در یک مجموعهٔ RRSet قرار می‌گیرند.[2] یک مجموعهٔ RRSet ممکن است شامل چندین رکورد A (رکوردهای آدرس IP) باشد چراکه مثلاً یک سرویس‌دهنده ممکن است دارای یک آدرس IP اولیه و یک آدرس ثانویه باشد. در هر مجموعهٔ RRSet چندین رکورد نوع جدید (که در ادامه بررسی خواهند شد) قرار می‌گیرد. همچنین برای هر مجموعهٔ RRSet یک رشتهٔ Hash (که صحت آن مجموعه را تائید می‌کند) به یکی از روشهای معمول مثل SHA-1 یا MD5 محاسبه می‌شود و نهایتاً با استفاده از کلید خصوصی صاحب آن ناحیه (Zone) به روشی مثل RSA رمزنگاری می‌گردد. کوچکترین واحد اطلاعات که برای مشتری ارسال می‌شود یک مجموعهٔ امضاء شدهٔ RRSet است. پس از دریافت RRSet امضاء شده، مشتری (Client) می‌تواند بررسی کند که آیا واقعاً با کلید خصوصی ناحیهٔ مبداء امضا شده است؟ اگر امضای آن منطبق بود داده‌ها پذیرفته می‌شوند. از آنجایی که هر مجموعهٔ RRSet امضای خودش را به همراه دارد می‌تواند در هر جایی ذخیره (cache) شود، حتی بر روی سرویس دهنده‌های غیرقابل اعتماد؛ بدون آن که خدشه‌ای به امنیت آنها وارد گردد.

DNSsec چندین نوع رکورد جدید معرفی کرده است. اولین آنها رکورد نوع KEY است. این رکورد کلید عمومی یک «ناحیه» (Zone) یا کاربر یا ماشین میزبان (Host) و همچنین الگوریتم رمزنگاری بکار رفته برای امضاء ، پروتکل انتقال و تعدادی مشخصهٔ دیگر را نگهداری می‌کند. کلید عمومی به صورت آشکار (رمزنشده) ذخیره می‌شود. از گواهینامه‌های X.509 به دلیل حجم بزرگ استفاده نشده است. در رکورد نوع KEY فیلد مشخص کنندهٔ الگوریتم، برای امضاهای مبتنی بر MD5/RSA به عدد ۱ تنظیم می‌شود (انتخاب ارجح) و مقادیر دیگر، ترکیب الگوریتمهای مورد نظر را مشخص می‌کند. فیلد پروتکل می‌تواند استفاده از IPsec یا هر پروتکل امنیتی دیگر را (در صورت وجود) تعیین نماید.

دومین نوع جدید رکوردها، رکورد SIG است. این رکورد، رشتهٔ Hash امضاء شدهٔ یک ناحیه (Zone) را

---

۱. Originating Zone
۲. در خصوص این واژه‌ها به فصل هفتم مراجعه کنید. -م.

مشخص می‌نماید. (رشتهٔ Hash براساس الگوریتمی که در رکورد KEY مشخص شده، امضاء می‌گردد.) این امضاء بر روی تمام رکوردهای یک مجموعهٔ RRSet شامل رکوردهای KEY (به استثنای خودش)، اعمال می‌شود. رکورد SIG، زمان شروع اعتبار و زمان انقضای امضاء و همچنین نام امضاءکننده و چند آیتم دیگر را نیز تعیین می‌کند.

طراحی DNSsec به گونه‌ای است که می‌توان کلیدهای خصوصی را به صورت جدا و در ماشینی غیرمتصل به شبکه (به صورت offline) نگهداری کرد. هر روز یا هر دو روز یکبار، محتویات پایگاه دادهٔ DNS به صورت دستی (مثلاً توسط CD-ROM) بر روی یک ماشین غیرمتصل به شبکه که کلیدهای خصوصی را نگه می‌دارد منتقل می‌گردد. در آنجا تمام مجموعه‌های RRSet امضاء شده و برای هر مجموعه، یک رکورد SIG (امضاء) تولید و نتیجه توسط CD-ROM به ماشین اصلی برگردانده می‌شود. بدین ترتیب می‌توان کلیدهای خصوصی را بر روی یک CD-ROM ذخیره کرد و CD-ROM به جز در مواقعی که باید مجموعه‌های RRSet را امضاء کرد در جای مطمئنی نگهداری شود. پس از امضاء، حافظه و دیسک آن ماشین کاملاً پاک شده و CD-ROM به محل امن خود برگردانده می‌شود. در این روال، امنیت الکترونیکی به امنیت فیزیکی کاهش می‌یابد و پیچیدگی چندانی ندارد.

این روش که در آن مجموعه‌های RRSet از قبل امضاء می‌شوند، سرعت فرآیند پاسخ به تقاضاها را بشدت افزایش می‌دهد چرا که لازم نیست در حین پاسخ و ارسال هیچگونه عمل رمزنگاری انجام شود.

وقتی ماشین مشتری، یک مجموعهٔ RRSet امضاء شده را دریافت می‌کند ابتدا باید کلید عمومی آن ناحیه را اعمال کرده و رشتهٔ Hash را بدست بیاورد؛ سپس خودش رشتهٔ Hash را مستقلاً محاسبه کرده و این دو مقدار را با هم مقایسه کند. اگر این دو مقدار یکسان بود، داده‌ها معتبر فرض می‌شود. این روال یک سؤال ایجاد می‌کند و آن هم این که مشتری چگونه کلید عمومی هر ناحیه را بدست آورده و بدان اعتماد کند؟ یک راه حل آن است که این کلیدها از یک سرویس‌دهندهٔ معتبر و مورد اعتماد و براساس یک اتصال امن نظیر IPsec اخذ شود.

با این وجود، در عمل فرض شده که ماشینهای مشتری به گونه‌ای پیکربندی شده‌اند که کلیدهای عمومی حوزه‌های سطح بالا را به صورت پیش فرض در اختیار دارند. حال اگر آلیس بخواهد به وب‌سایتِ باب مراجعه کند می‌تواند از DNS بخواهد که مجموعهٔ RRSet متناظر با bob.com را در اختیار او بگذارد که در آن آدرس IP و همچنین رکورد KEY (شامل کلید عمومی باب) مشخص شده است. مجموعهٔ RRSet توسط مسئول حوزهٔ سطح بالای com. امضاء می‌شود لذا آلیس می‌تواند اعتبار آن را بررسی کند زیرا کلید عمومی حوزهٔ com. را همه می‌دانند و نیازی به سؤال نیست. یک مثال از رکوردهایی که می‌تواند در یک مجموعهٔ RRSet وجود داشته باشد در شکل ۸–۴۸ نشان داده شده است.

حال با در اختیار داشتن نسخه‌ای معتبر از کلید عمومی باب، آلیس می‌تواند از سرویس دهندهٔ DNS متعلق به باب، در خصوص آدرس IP معادل با www.bob.com سؤال کند. اگر ترودی به نحوی برنامه‌ریزی کند که تعدادی RRSet را به صورت جعلی تشکیل داده و در حافظهٔ نهان یک سرویس دهندهٔ DNSsec تزریق نماید، آلیس قادر

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

شکل ۸–۴۸. مثالی از رکوردهای RRSet برای نام حوزهٔ bob.com. رکورد KEY حاوی کلید عمومی باب است. رکورد SIG حاوی رشتهٔ HASH امضاء شده توسط سرویس‌دهندهٔ سطح بالای com. است که برای رکوردهای نوع A و KEY محاسبه شده تا بتوان اعتبار و هویت این رکوردها را بررسی کرد.

خواهد بود به دلیل فقدان اعتبار، این موضوع را کشف کند چراکه رکورد SIG (امضاء) با مجموعهٔ RRSet مطابقت ندارد و غلط است.

با این وجود DNSsec یک مکانیزم مبتنی بر رمزنگاری برای تطبیق بسته‌های پاسخ متناظر با تقاضاهای ارسالی ارائه کرده است تا از پی‌ریزی حملهٔ DNS Spoofing توسط ترودی (مبتنی بر شکل ۸-۴۷) جلوگیری شود. در این مکانیزم اختیاری (که antispoof نام دارد) به هر بستهٔ پاسخ یک رشتهٔ Hash اضافه می‌شود که این Hash از بستهٔ تقاضا استخراج شده و سپس با کلید خصوصی پاسخ دهنده، امضا (رمز) می‌شود. از آنجایی که ترودی کلید خصوصی سرویس دهندهٔ سطح بالای com. را نمی‌داند لذا قطعاً نخواهد توانست پاسخ ارسالی توسط این سرویس دهنده به ISP آلیس را جعل نماید. البته او می‌تواند بستهٔ جعلی خود را برای آلیس بفرستد ولیکن به دلیل امضای اشتباهی که دارد، حذف خواهد شد.

DNSsec از چندین رکورد دیگر نیز حمایت می‌کند. به عنوان مثال رکورد CERT می‌تواند برای ذخیره و نگهداری گواهینامه‌هایی مثل X.509 مورد استفاده قرار بگیرد. این رکورد بدان دلیل عرضه شده که برخی افراد علاقمندند DNS خود را در ساختار PKI[1] قرار بدهند؛ اگرچه هنوز در عمل چنین کاری انجام نشده است. در اینجا توضیحات خود در خصوص DNSsec را به پایان می‌رسانیم. برای تفصیل بیشتر از RFC 2535 کمک بگیرید.

«اسامی خودگواهی» (Self-Certifying Names)

استفاده از سرویس‌دهندهٔ امن DNS (DNSsec) تنها راه اطمینان کردن به اسامی نیست. یک راهکار کاملاً متفاوت در Secure File System (سیستم مطمئن فایل) بکار گرفته می‌شود. (Mazieres et al., 1999) پژوهشگران این پروژه، یک سیستم فایل مطمئن، جهانی و قابل گسترش طراحی کردند، بدون آنکه نیاز به تغییری در DNS باشد و حتی بدون استفاده از گواهینامه‌های دیجیتالی یا بدون وجود هیچگونه PKI کار کند. در این بخش نشان خواهیم داد که چگونه می‌توان نظریات آنان را در وب اعمال کرد. بر این اساس، در توضیحات این بخش به جای استفاده از اصطلاحات و واژه‌های بکار رفته در مقالهٔ آنان، از اصطلاحات وب استفاده شده است؛ ولیکن برای احتراز از پیچیده شدن احتمالی بحث، باید اشاره کرد که این ساختار هنوز در سیستم وب بکار گرفته نشده ولی برای رسیدن به امنیت بالا در وب، این روش ممکن و مناسب است اگرچه قبل از معرفی و بکارگیری، باید تغییراتی اساسی در نرم‌افزارهای مرتبط با وب ایجاد شود.

برای شروع فرض می‌کنیم که هر سرویس‌دهندهٔ وب دارای یک جفت کلید عمومی و خصوصی است. دلیل این فرض آنست که در هر URL یک رشتهٔ Hash برای بررسی صحت نام سرویس دهنده و صحت کلید عمومی آن، (به عنوان بخشی از URL) درج می‌شود. به عنوان مثال در شکل ۸-۴۹، URL یک فایل تصویر متعلق به باب را می‌بینیم. این آدرس طبق معمول با گزینهٔ http شروع شده که نام پروتکل[2] را تعیین می‌کند، سپس در ادامه، آدرس DNS سرویس دهندهٔ باب (یعنی *www.bob.com*) درج شده است. بعد از آن یک علامت کالون (:) و سپس ۳۲ کاراکتر به عنوان رشتهٔ Hash اضافه شده است. در انتهای این رشته، باز هم طبق روش معمول، نام و موقعیت فایل می‌آید. به غیر از رشتهٔ Hash ، بقیهٔ URL استاندارد و معمولی است. با وجود رشتهٔ Hash، این URL اصطلاحاً «**خودگواهی**» می‌شود؛ (خودش صحت خود را تائید می‌کند).

سؤال بدیهی آن است که: رشته Hash برای چیست؟ پس از الحاق نام نمادین و کلید عمومی یک سرویس‌دهنده به یکدیگر، الگوریتم SHA-1 بر روی آن اعمال می‌شود تا رشتهٔ ۱۶۰ بیتی Hash بدست آید. در

۱. Public Key Infrastructure

۲. Protocol یا Schema

Server | SHA-1 (Server, Server's Public key) | File name

http://www.bob.com:2g5hd8bfjkc7mf6hg8dgany23xds4pe6/photos/bob.jpg

شکل ۸-۴۹. یک «URL خودگواهی» شامل رشتهٔ HASH از نام و کلید عمومی سرویس‌دهنده.

الگوی فوق رشته Hash به صورت دنباله‌ای از ۳۲ رقم و حروف کوچک نشان داده شده است با این استثنا که از حروف 'l' و 'o' و ارقام '1' و '0' به دلیل مشابهت در شکل نمایش و احتمال خطا در حین وارد کردن URL صرفنظر شده است، بنابراین جمعاً ۳۲ حرف و رقم باقی می‌ماند؛ (۲۴ حرف کوچک و ۸ رقم). با این سی و دو کاراکتر می‌توان پنج بیت را کدگذاری کرد. بدین ترتیب با یک رشتهٔ ۳۲ کاراکتری می‌توان جمعاً ۱۶۰ بیت رشته SHA-1 را کُد نمود؛ (۳۲ کاراکتر هر کاراکتر معادل ۵ بیت) البته در واقع، الزامی در استفاده از Hash نیست و می‌توان به جای آن خود کلید عمومی را قرار داد؛ حُسن استفاده از رشتهٔ Hash آنست که طول نام (که رشتهٔ Hash را در بر می‌گیرد) کاهش یابد.

ساده‌ترین روش (ولی در عین حال نامناسب‌ترین روش از لحاظ سهولت) برای مراجعه به فایل تصویر باب، آنست که آلیس رشتهٔ شکل ۸-۴۹ را در مرورگر خود درج نماید. مرورگر پیامی برای سرویس‌دهندهٔ وب متعلق به باب فرستاده و کلید عمومی او را مطالبه می‌کند. وقتی کلید عمومی باب دریافت شد مرورگر، نام سرویس دهنده و کلید عمومی را بهم چسبانده و الگوریتم Hash را بر روی آن اعمال می‌کند. اگر رشتهٔ Hash حاصل‌شده با رشتهٔ Hash سی و دو کاراکتری در URL مطابقت داشت، مرورگر مطمئن خواهد شد که کلید عمومی ارسالی واقعاً متعلق به باب است. حتی اگر ترودی بتواند در میانهٔ راه این تقاضا را دریافت کرده و یک پاسخ جعلی برگرداند نمی‌تواند هیچ کلید عمومی که رشتهٔ Hash آن با رشتهٔ درج شده در URL مطابقت داشته باشد پیدا کند لذا هر گونه دخالت او در جعل پاسخ، به سادگی کشف خواهد شد. پس از دریافت کلید عمومی باب، می‌توان آن را برای استفاده‌های آتی در حافظهٔ نهان (cache) ذخیره کرد.

حال آلیس باید بررسی کند که آیا باب واقعاً کلید خصوصی خود را (متناظر با کلید عمومی ارسال شده) در اختیار دارد یا خیر؟ او یک پیام می‌سازد که در برگیرنده یک کلید نشست AES، یک عدد تصادفی (nonce) و یک مهر زمان است. سپس او این پیام را با کلید عمومی باب رمز کرده و آن را برای او می‌فرستد. از آنجایی که فقط باب کلید خصوصی متناظر با این کلید عمومی را در اختیار دارد بنابراین تنها او قادر به رمزگشایی آن و بازگرداندن عدد تصادفی (nonce) با کلید AES است. در صورتی که عدد تصادفی که با کلید AES رمز و برگردانده شده با آنچه که توسط آلیس ارسال شده یکسان باشد، او می‌تواند مطمئن شود که با باب صحبت می‌کند. همچنین در اینجا آلیس و باب دارای یک کلید نشست AES مشترک هستند که می‌توانند برای تقاضاهای GET بعدی و پاسخهای ارسالی از آن استفاده کنند.

پس از آن که آلیس فایل تصویر متعلق به باب (یا هر صفحهٔ وب دیگر) را بدست آورد می‌تواند آدرس URL آن را در دفترچهٔ یادداشت مرورگر خود ذخیره نماید، بدین ترتیب او مجبور نخواهد بود در آینده، URL کامل را تایپ کند. بعلاوه URLهای جاسازی شده در درون صفحات وب نیز می‌توانند از نوع **«خودگواهی»** باشند، بنابراین فقط با یک کلیک می‌توان از آنها به صورت معمولی استفاده کرد؛ البته بشرطی که صفحهٔ وب برگشتی واقعی و دست نخورده باشد. یک روش دیگر برای آن که نیازی به درج URLهای **«خودگواهی»** نباشد آنست که این اسامی را از طریق یک سرویس‌دهندهٔ مورد اعتماد (که دارای گواهینامهٔ X.509 امضا شده توسط CA[1] باشد) و

۱. Certificate Authority

با برقراری یک اتصال امن (Secure Connection) دریافت نمائیم.

روش دیگر برای دریافت URLهای خودگواهی آنست که با وارد کردن «URL خودگواهی» یک موتور جستجوی مورد اعتماد و مطمئن (فقط برای یکبار) به آن متصل شده و به روش تشریح شده در بالا یک ارتباط مطمئن و احراز هویت شده با آن موتور جستجوی امن برقرار نمائیم. حال می‌توان این موتور جستجو را در خصوص یک URL مورد سؤال قرار داد و طبعاً نتیجهٔ کار، یک صفحهٔ وب امضاء شده خواهد بود که درون آن تمام URLها از نوع «خودگواهی» هستند و می‌توان بر روی آنها کلیک کرد، بدون آن که نیازی به درج رشته‌های طولانی باشد.

حال ببینیم این روش چگونه می‌تواند در مقابل حملهٔ DNS Spoofing که توسط ترودی انجام می‌شود مقاومت نماید: اگر ترودی به نحوی برنامه‌ریزی کند تا حافظهٔ نهان ISP متعلق به آلیس آلوده شود، تقاضای آلیس برای دریافت یک صفحهٔ وب، به جای آن که توسط باب دریافت شود تحویل ترودی خواهد شد. حال طبق پروتکل فوق، مرورگر آلیس در اولین پیام، از ترودی می‌خواهد که کلید عمومی خود را ارائه بدهد. اگر ترودی کلید عمومی خود را برگرداند، آلیس فوراً حمله را کشف خواهد کرد چراکه رشتهٔ Hash موجود در URL خودگواهی (که مبتنی بر کلید عمومی باب است)، با رشتهٔ Hash محاسبه شده به روش SHA-1، منطبق نیست. اگر ترودی کلید عمومی باب را ارائه بدهد، آلیس قادر نخواهد بود حمله را کشف کند ولیکن از آنجایی که پیامهای بعدی را براساس کلید عمومی باب رمزنگاری و ارسال می‌کند لذا ترودی پس از دریافت این پیامها هیچ راهی برای رمزگشایی و استخراج کلید AES و عدد تصادفی درون پیام نخواهد داشت. در اینجا اگرچه حملهٔ DNS Spoofing سودی برای ترودی (جهت بهره‌برداری از اطلاعات ارسالی) ندارد ولیکن می‌تواند برای حملهٔ «اخلال در سرویس‌دهی» (Denial of Service) مؤثر باشد، زیرا ترودی کاری کرده که آلیس نتواند با باب ارتباط برقرار کند.

۸-۹-۳ SSL: لایهٔ سوکتهای امن

نامگذاری مطمئن» اگرچه شروع خوبی است ولیکن برای امنیت وب امکانات بسیار زیادتری وجود دارد. گام بعدی در امنیت وب برقراری اتصال مطمئن (Secure Connection) است. حال ببینیم اتصالات امن چگونه بوجود می‌آیند. در بدو زمانی که وب در صحنهٔ اجتماع پدیدار گردید، از آن فقط برای توزیع صفحات وب ایستا[1] استفاده می‌شد. با این وجود در اندک زمانی، بسیاری از شرکتها به صرافت افتادند تا از وب برای معاملات اقتصادی مثل داد و ستد کالا با کارتهای اعتباری، بانکداری Online و خرید و فروش الکترونیکی سهام استفاده کنند. این نوع کاربردها نیاز به ایجاد **«اتصال امن»** (Secure Connection) را دامن زدند. در سال ۱۹۹۵ شرکت **نت‌اسکیپ** (Netscape) عرضه کنندهٔ مرورگر مهم دنیا، با معرفی یک بستهٔ امنیتی به نام SSL (Secure Socket Layer) به این نیاز پاسخ داد. امروزه از این نرم‌افزار و پروتکل معرفی شدهٔ آن، بطور فزاینده‌ای استفاده می‌شود و حتی مرورگر IE مایکروسافت نیز از آن حمایت می‌کند، لذا بجاست که آنرا با اندکی شرح بیشتر بررسی کنیم.

«SSL یک اتصال مطمئن با مشخصات زیر، بین دو سوکت ایجاد می‌نماید:

۱. امکان مذاکره و توافق پارامترها بین سرویس دهنده و مشتری
۲. احراز هویت سرویس دهنده و مشتری بطور مستقل و مجزا
۳. مخابرهٔ سرّی و رمزنگاری شدهٔ داده‌ها
۴. مراقبت از صحت و سلامت داده‌ها

---

۱. صفحاتی که فقط ارائه دهندهٔ اطلاعات هستند و هیچگونه دریافت اطلاعات ندارند. ـم

قبلاً هر یک از موضوعات فوق را به صورت جداگانه بررسی کرده‌ایم و نیازی به شرح مبسوط آنها نیست.

محل قرار گرفتن لایهٔ SSL در پشتهٔ پروتکلی TCP/IP، در شکل ۸-۵۰ نشان داده شده است. در حقیقت SSL یک لایهٔ جدید بین لایهٔ کاربرد و لایهٔ انتقال است که تقاضاهای مرورگر را گرفته و آنها را از طریق لایهٔ TCP برای سرویس دهنده ارسال می‌کند. پس از آن که یک اتصال مطمئن ایجاد شد، مهمترین وظیفهٔ SSL ، انجام عملیات فشرده‌سازی و رمزنگاری اطلاعات (و بالعکس) خواهد بود. وقتی از HTTP بر روی SSL استفاده می‌شود، اصطلاحاً آنرا HTTPS (Secure HTTP) می‌نامند، هرچند هیچ تفاوتی با پروتکل استاندارد HTTP ندارد؛ فقط در لایهٔ زیرین تغییراتی ایجاد شده است. البته برخی اوقات به جای پورت استاندارد ۸۰، HTTPS بر روی پورت جدید ۴۴۳ در دسترس قرار می‌گیرد. از آنجایی که SSL یک لایهٔ مستقل بر روی TCP است لذا استفاده از آن به مرورگرهای وب محدود نشده و دیگر برنامه‌ها نیز می‌توانند از امکانات آن استفاده کنند ولیکن عمومی‌ترین کاربرد آن در وب است.

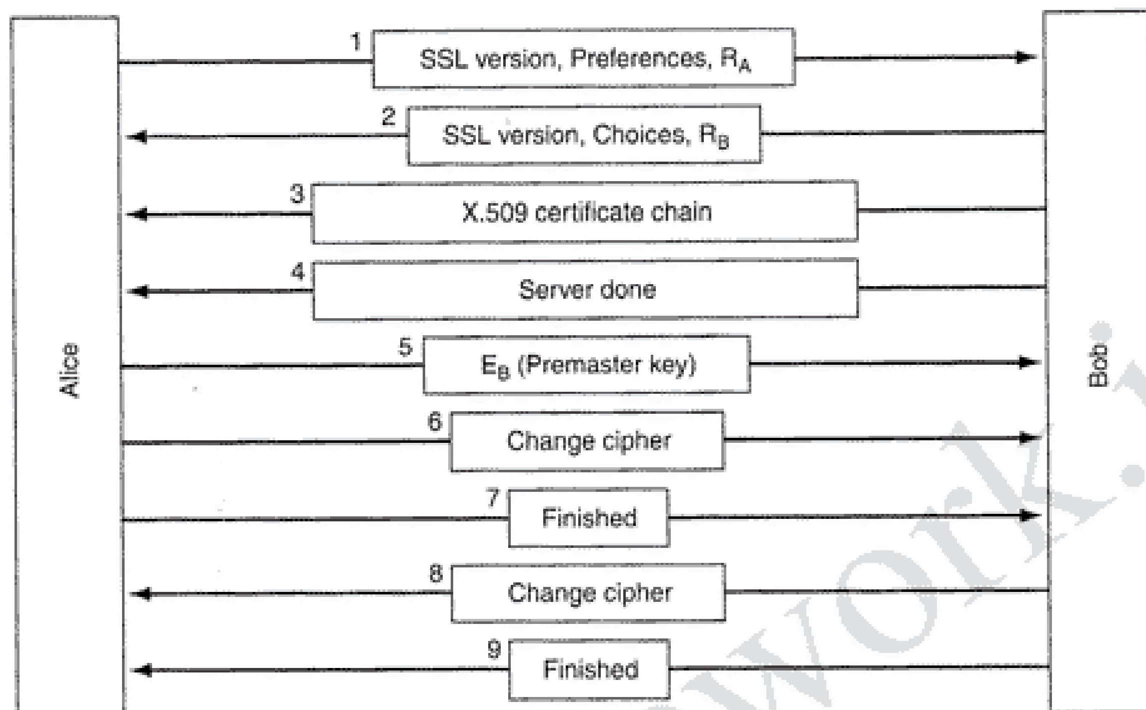
Application (HTTP)
Security (SSL)
Transport (TCP)
Network (IP)
Data link (PPP)
Physical (modem, ADSL, cable TV)

شکل ۸-۵۰. لایه‌ها (و پروتکلها) برای یک یک کاربر خانگی که از طریق SSL به مرور وب می‌پردازد.

پروتکل SSL از چندین نسخهٔ مختلف، گذر کرده و متکامل شده است. در زیر ما فقط نسخهٔ ۳ آنرا بررسی می‌کنیم چراکه بطور فزآینده‌ای از آن استفاده می‌شود و نسخه‌های قبلی عملاً منسوخ شده‌اند. SSL از چندین الگوریتم و گزینه‌های مختلف حمایت می‌کند. گزینه‌ها شامل استفاده یا عدم استفاده از فشرده‌سازی ، تعیین نوع الگوریتم رمزنگاری و مواردی در خصوص محدودیت صادرات محصولات مبتنی بر رمزنگاری است. آخرین مورد یعنی گزینه‌های مرتبط با محدودیتهای صادرات بدان منظور بوده تا از امکانات پیشرفته رمزنگاری [که طبق قوانین ایالات متحده، صدور آنها به خارج محدود است] فقط زمانی استفاده شود که طرفین یک اتصال، صرفاً در ایالات متحده مستقر باشند. در بقیهٔ موارد طول کلید به ۴۰ بیت محدود شده است که بسیاری از رمزنگاران این طول کلید را یک شوخی تلقی می‌کنند. شرکت نت‌اسکیپ برای آن که بتواند مجوز صدور محصول خود را از دولت ایالات متحده اخذ کند مجبور بود این محدودیت را اعمال نماید.

SSL از دو «**زیرپروتکل**» (subprotocol) تشکیل شده است: یکی برای ایجاد اتصال امن و دیگری برای بکارگیری از آن جهت مبادله اطلاعات. اجازه بدهید در ابتدا ببینیم که چگونه اتصالات امن بوجود می‌آید. زیرپروتکل ایجاد اتصال، در شکل ۸-۵۱ نشان داده شده است. وقتی آلیس تقاضای برقراری یک ارتباط امن با باب را می‌دهد، این زیرپروتکل کار خود را با ارسال پیام ۱ شروع می‌کند. این پیام نسخهٔ SSL مورد استفاده توسط آلیس و همچنین گزینه‌های انتخابی او نظیر تمایل به فشرده‌سازی و الگوریتم مورد نظر او جهت رمزنگاری را مشخص می‌کند. این پیام همچنین شامل یک عدد تصادفی بزرگ R_A است که به عنوان رشته چالش (nonce) بعداً از آن استفاده خواهد شد.

حال نوبت باب است. در پیام ۲، باب از بین گزینه‌های پیشنهادی که آلیس از آنها حمایت می‌کند، گزینه‌هایی را انتخاب کرده و این انتخابها را به همراه عدد تصادفی R_B و نسخهٔ SSL مورد استفاده، برای آلیس می‌فرستد. سپس



شکل ۸-۵۱. نسخهٔ ساده شده از زیرپروتکل برقراری اتصال در SSL.

در پیام سوم باب گواهینامهٔ دیجیتالی خود را که در آن کلید عمومی او درج شده، برای آلیس می‌فرستد. اگر این گواهینامه توسط یک مرکز معتبر، امضا نشده باشد، باب زنجیره‌ای از گواهینامه‌ها را که نهایتاً به یک مرکز معتبر ختم می‌شود، برای آلیس می‌فرستد. تمام مرورگرها و از جمله مرورگر آلیس، دارای حدود صد کلید عمومی از مراکز شناخته شده و مجاز گواهی امضاء هستند که باب باید زنجیرهٔ گواهینامه‌ها را به نحوی برای آلیس بفرستد تا نهایتاً به یکی از این صد مرکز ختم شود.[1] بدین ترتیب آلیس قادر خواهد بود تا صحت کلید عمومی باب را بررسی نماید. در اینجا باب ممکن است پیامهای دیگری برای آلیس ارسال کند (مثل تقاضای گواهینامهٔ دیجیتالی آلیس). پس از انجام این عملیات او پیام ۴ را برای آلیس فرستاده و به او اعلام می‌کند که نوبت اوست.

آلیس در پیام ۵، با انتخاب یک **«شاه‌کلید»** اولیه ۳۸۴بیتی و ارسال آن برای باب به او پاسخ می‌دهد در حالی که این شاه‌کلید با کلید عمومی باب رمزنگاری شده است. کلید اصلی نشست که بعداً برای رمزنگاری اطلاعات استفاده خواهد شد از ترکیب شاه کلید و R_A و R_B به روش پیچیده‌ای استخراج می‌گردد. پس از دریافت پیام ۵، آلیس و باب هر دو قادرند کلید اصلی نشست را محاسبه نمایند. به همین دلیل آلیس در پیام ۶ به باب اعلام می‌کند که از سیستم رمز جدید (با کلید جدید) استفاده کند. در پیام ۷، خاتمهٔ زیرپروتکل ایجاد اتصال اعلام می‌شود. باب نیز در پاسخ، آلیس را تایید و ختم زیرپروتکل را اعلام می‌کند. (پیامهای ۸ و ۹)

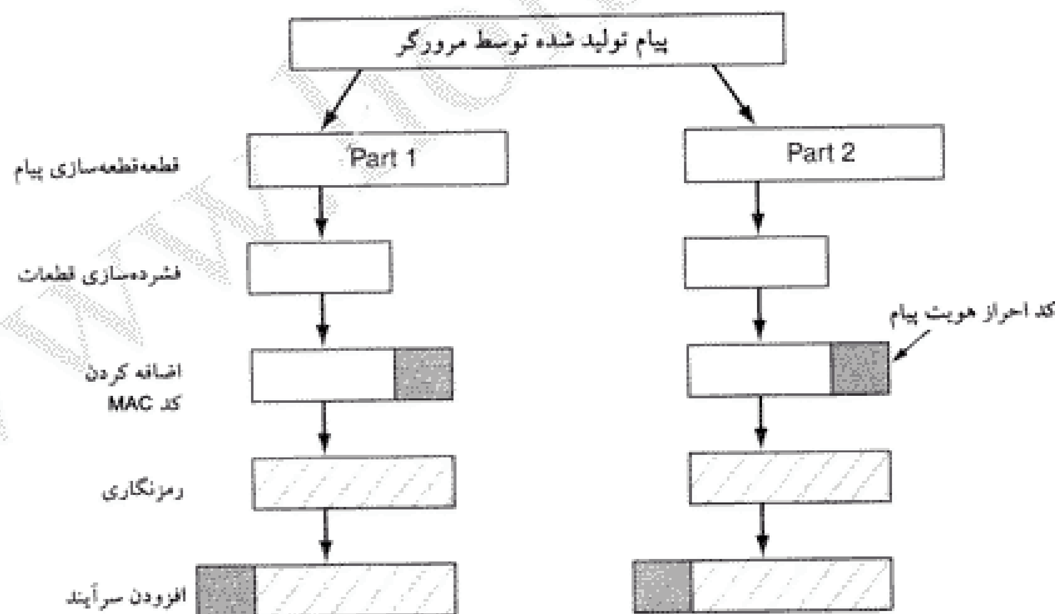
در اینجا اگرچه آلیس از باب مطمئن است ولی باب نمی‌داند که آلیس کیست (مگر آن که آلیس دارای یک کلید عمومی و یک گواهینامه دیجیتالی معتبر باشد.) بنابراین اولین پیامی که ممکن است باب پس از ایجاد اتصال برای آلیس بفرستد آن است که از او تقاضا کند با ارائه نام ورود و کلمهٔ عبور، Login نماید. پروتکل لازم برای عملیات Login، خارج از محدودهٔ این کتاب است. پس از تکمیل عملیات Login، انتقال اطلاعات می‌تواند آغاز شود.

به گونه‌ای که قبلا اشاره شد، SSL از چندین الگوریتم رمزنگاری حمایت می‌کند. در قدرتمندترین روش،

۱. برای آشنایی با زنجیرهٔ گواهینامه‌های دیجیتالی به بخش ۸-۵-۳ مراجعه کنید

برای رمزنگاری از triple DES با سه کلید مجزا و برای بررسی صحت پیامها از SHA-1 استفاده می‌شود. ترکیب این دو پروتکل نسبتاً کُند عمل می‌کند و اغلب برای عملیات بانکداری یا کاربردهایی که در آنها امنیت بالا حیاتی است مورد استفاده قرار می‌گیرد. برای عملیات معمولی تجارت الکترونیکی، از رمزنگاری RC4 با کلید ۱۲۸ بیتی و روش MD5 برای تائید صحت اطلاعات استفاده می‌شود. RC4 کلید ۱۲۸ بیتی را به عنوان نقطهٔ شروع گرفته و برای استفاده در الگوریتم، آنرا به یک عدد بزرگ بسط می‌دهد و سپس از آن برای تولید Keystream بهره می‌گیرد. این Keystream (به نحوی که در بخش ۸-۲-۳ تشریح شد) با داده‌های ارسالی ، XOR می‌شود تا متن رمز شده (به روش Stream Cipher) بدست بیاید؛ (شکل ۸-۵۲). نسخهٔ صادراتی SSL نیز از کلیدهای ۱۲۸ بیتی RC4 استفاده می‌کند ولیکن ۸۸ بیت از آن عمومی و آشکار است تا بتوان رمز آن را براحتی شکست. (زیرا صدور محصولاتِ SSL از ایالات متحده، فقط با کلید ۴۰ بیتی مجاز است.)

به نحوی که در شکل ۸-۵۲ نشان داده شده، برای انتقال واقعی اطلاعات، از زیرپروتکل دوم SSL استفاده می‌شود. پیامهای ارسالی از مرورگر، ابتدا به قطعات ۱۶ کیلوبایتی شکسته می‌شود. اگر گزینهٔ فشرده‌سازی فعال شده باشد، هر قطعه به صورت مستقل فشرده می‌شود. سپس یک کلید سرّی که از ترکیب دو عدد تصادفی (nonce) و شاه‌کلید اولیه بدست می‌آید (موقتاً) به متن فشرده‌شده ضمیمه می‌شود و براساس الگوریتم مورد توافق (معمولاً MD5)، از آن یک رشته Hash استخراج می‌گردد. این رشتهٔ Hash به عنوان MAC[1] (کد تائید صحت پیام) به قطعهٔ فشرده‌شده ضمیمه می‌شود. سپس مجموعهٔ این دو با استفاده از الگوریتم متقارن مورد توافق (معمولاً با XOR کردن داده‌ها با Keystreamهای مبتنی بر الگوریتم RC4)، رمزنگاری می‌شود. نهایتاً به قطعهٔ حاصل یک سرآیند اضافه شده و نتیجه بر روی اتصال TCP ارسال می‌گردد.



شکل ۸-۵۲. انتقال داده با استفاده از SSL.

در استفاده از RC4 یک مورد احتیاط وجود دارد: از آنجایی که اثبات شده در الگوریتم RC4 ، کلیدهای ضعیف و نامناسبی وجود دارند که از طریق آنها می‌توان رمز متن را شکست، لذا امنیت SSL وقتی که در آن از RC4 استفاده شده بسیار متزلزل و شکننده است. مرورگرهایی به کاربر اجازه می‌دهند شخصاً الگوریتم مورد

۱. Message Authentication Code

نظرش را انتخاب نماید باید به نحوی پیکربندی شوند تا همیشه از روش رمزنگاری Triple DES با کلیدهای ۱۶۸ بیتی و روش SHA-1 استفاده کنند اگرچه این ترکیب بسیار کندتر از ترکیب RC4 و MD5 عمل می‌کند.

اشکال دیگر SSL آن است که طرفین ارتباط ممکن است گواهینامهٔ دیجیتالی نداشته باشند یا حتی در صورت دارا بودن گواهینامه، همیشه صحّت و تطابق کلیدهای مورد استفاده را بررسی نکنند.

در سال ۱۹۹۶ شرکت نت‌اسکیپ، SSL را برای استانداردسازی به IETF ارائه کرد. نتیجهٔ کار TLS (Transprot Layer Security) بود که در RFC 2246 تشریح شده است.

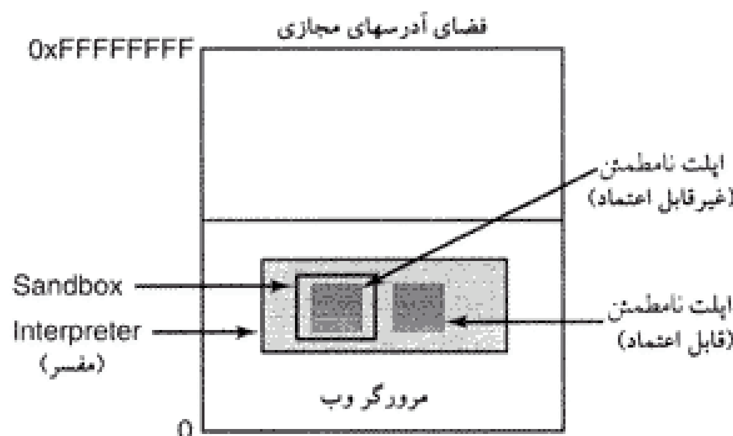
تغییراتی که در SSL ایجاد شد نسبتاً کم بود ولیکن همین تغییرات ناچیز کافی بود تا SSL نسخهٔ ۳ با TLS سازگار نباشد. به عنوان مثال روشی که براساس آن از شاه‌کلید اولیه و اعداد تصادفی (nonce)، کلید نشست استخراج می‌شود تغییر داده شد تا کلید قوی‌تر و محکم‌تر شود. (رمزشکنی آن سخت‌تر شود.) TLS گاهی به عنوان نسخهٔ 3.1 از SSL نیز نامیده می‌شود. اولین پیاده‌سازی TLS در سال ۱۹۹۹ عرضه شد ولی هنوز مشخص نیست که عملاً چه زمانی جایگزین SSL خواهد شد، هرچند از SSL نسبتاً قویتر است. البته اشکال ضعیف بودن کلیدهای RC4 هنوز هم در TLS وجود دارد.

۸-۹-۴ امنیت کدهای متحرک

«نامگذاری مطمئن» و «ایجاد اتصالات امن»، دو زمینه مهم و مرتبط با امنیت وب است ولی هنوز موارد متعدد دیگری وجود دارد. در روزهای اولیه، صفحات وب فقط فایلهای HTML ایستا بودند و درون آنها هیچگونه کد اجرایی وجود نداشت. امروزه اغلب صفحات وب، در برگیرندهٔ برنامه‌های کوچک اجرایی شامل اپلت‌های جاوا، کنترلهای Active X و اسکریپتهای جاوا هستند. دریافت و اجرای چنین کدهای متحرکی، خطرات امنیتی بسیار عظیمی دارد لذا باید روشهای مختلفی طراحی و ابداع شود تا این مخاطرات کاهش یابد. در این بخش برخی از موارد مرتبط با کدهای متحرک و روشهای برخورد با مخاطرات آن را بطور اجمالی مرور خواهیم کرد.

امنیت اپلتهای جاوا

اپلتهای جاوا، برنامه‌های کوچک جاوا هستند که برای اجرا بر روی یک «ماشین مجازی مبتنی بر پشته» (Stack Oriented) به نام JVM (Java Virtual Machine) ترجمه (کامپایل) می‌شوند. این برنامه‌ها می‌توانند درون یک صفحهٔ وب قرار گرفته و به همراه آن صفحه، بارگذاری و اجرا شوند. به گونه‌ای که در شکل ۸-۵۳ نشان داده شده است پس از بارگذاری صفحهٔ وب، اپلتها جهت اجرا تحویل مفسر JVM (تعبیه شده در درون مرورگر) می‌شوند.



شکل ۸-۵۳. اپلتها می‌توانند توسط مرورگر تفسیر و اجرا شوند.

یک مزیت اجرای کدهای مفسّری (Interpreted Code) در مقایسه با کدهای ترجمه شده (Compiled Code) آن است که هر دستورالعمل قبل از اجرا توسط مفسر بررسی می‌شود. این قابلیت، فرصت بررسی اعتبار آدرس هر دستورالعمل را به مفسر فرمان می‌دهد. به علاوه هرگونه فراخوانی سیستمی (System Call) توسط مفسر دریافت و قبل از اجرا تفسیر می‌شود. چگونگی برخورد با این فراخوانی‌های سیستمی، جوهرهٔ سیاستهای امنیتی را تشکیل می‌دهد. به عنوان مثال هر گاه یک اپلت قابل اعتماد باشد (از دیسک محلی خودِ کامپیوتر دریافت و اجرا شده باشد) هر گونه فراخوانی سیستمی بدون چون و چرا اجرا می‌شود ولیکن اگر اپلت مورد اعتماد نباشد (از طریق اینترنت دریافت شده باشد) باید در یک شرایط کاملاً بسته و حفاظت شده (که Sandbox نامیده می‌شود) اجرا گردد تا بتوان بر رفتار آن، اعمال محدودیت کرد و از هر گونه تلاش برای دسترسی به منابع سیستم جلوگیری شود.

هر گاه یک اپلت تلاش کند تا از منابع سیستمی ماشین بهره بگیرد، فراخوانی سیستمی آن اپلت، به منظور تائید بلافاصله تحویل یک **«ناظر امنیت»** (Security Monitor) می‌شود. «ناظر امنیت» براساس سیاستهای امنیتی که به صورت محلی تدوین شده تصمیم می‌گیرد که آیا باید به آن فراخوانی اجازهٔ اجرا بدهد یا آنرا ردّ کند. بدین ترتیب فقط امکان دسترسی به برخی از منابع سیستمی وجود دارد نه همهٔ آنها. متأسفانه حقیقت آنست که این مدل امنیتی بد عمل می‌کند و اشکالات آن همیشه به ناگاه پدیدار می‌شود.

Active X

کنترلهای اکتیواکس، برنامه‌های اجرایی و دودویی برای ماشینهای پنتیوم هستند که می‌توانند درون صفحات وب جاسازی شوند. وقتی مرورگر به یکی از آنها بر می‌خورد، ابتدا بازرسی و آزمایشهایی انجام می‌شود تا ببینند آیا قابل اجرا است؛ در صورتی که بتواند این آزمون را با موفقیت بگذراند اجرا می‌شود. این کنترلها به هیچ‌وجه تفسیر (Interpret) و نظارت نمی‌شوند لذا به اندازهٔ برنامهٔ معمولی کاربران قدرت اجرایی دارند و می‌توانند آسیبهای بالقوه خطرناکی را به سیستم تحمیل کنند.

روشی که شرکت مایکروسافت برای تصمیم‌گیری در خصوص اجرا یا عدم اجرای کنترلهای اکتیواکس برگزیده است مبتنی بر روشی به نام **«امضای کد»** (Code Signing) است. هر کنترل اکتیواکس یک امضای دیجیتالی با خود به همراه دارد که در حقیقت یک رشتهٔ Hash از کد اجرایی است که توسط کلید خصوصی طراح آن، رمزنگاری و امضاء می‌شود. وقتی یک کنترل اکتیواکس ظاهر می‌شود، مرورگر ابتدا امضای آن را بررسی می‌کند تا مطمئن شود که در خلال انتقال دستکاری نشده است. اگر امضای آن صحیح بود مرورگر جدول داخلی خود را بررسی می‌کند تا ببیند آیا پدیدآورندهٔ آن برنامه قابل اعتماد است؟ یا آن که زنجیرهٔ گواهینامه‌های آن، به یک تولیدکننده مورد اعتماد ختم می‌شود یا خیر؟ اگر پدیدآورندهٔ اکتیواکس مورد اعتماد باشد، برنامه اجرا می‌شود، در غیر این صورت اجرا نخواهد شد. سیستمی که مایکروسافت برای بررسی کنترلهای اکتیواکس پیاده کرده به نام Authenticode مشهور است.

مقایسهٔ روشهای بکار رفته در جاوا و اکتیواکس مفید خواهد بود. در روش بکار رفته در جاوا هیچ تلاشی برای آن که مشخص شود چه کسی اپلت را نوشته، انجام نمی‌شود. در عوض یک «مفسر زمان اجراء» این اطمینان را فراهم می‌آورد که اپلت کارهایی را که صاحب ماشین اجازه نداده، انجام نمی‌دهد. برعکس، در اکتیواکس با اطمینان به امضای کدهای اجرایی، بر عملکرد و رفتار کُد در حال اجرا نظارت نمی‌شود. اگر آن برنامه از منبع قابل اعتماد دریافت شده و در حین گذر از شبکه دستکاری نشده باشد فوراً اجرا می‌شود. هیچ تلاشی نیز برای بررسی اینکه آیا کدهای برنامه مخرب هستند یا نه انجام نمی‌گیرد. اگر برنامه‌نویس اصلی اکتیواکس (که دارای گواهینامهٔ معتبر است و مورداعتماد تلقی می‌شود)، در نظر داشته باشد که کُد او کل دیسک سخت ماشین را نابود کرده و سپس

Flash ROM کامپیوتر را پاک کند تا دیگر بوت نشود، کُد او بی‌چون و چرا اجرا شده و کل کامپیوتر را نابود خواهد کرد. (مگر آن که گزینهٔ Active X در تنظیمات مرورگر غیرفعال شده باشد.)

بسیاری از افراد در خصوص اعتماد بـه شـرکتهای نـاشناختهٔ تـولید نـرم‌افـزار نگـران و بـدبین هـستند. یک برنامه‌نویس در سیاتل آمریکا برای آن که این اشکال را نشان بدهد یک شرکت نـرم‌افـزاری بـوجود آورد و یک گواهینامهٔ اعتماد برای خود تهیه کرد؛ انجام این کار ساده است. سپس یک کنترل اکتیواکس نوشت که کامپیوتر را خاموش (shutdown) می‌کرد. او این برنامه را در سطح وسیعی در اینترنت منتشر ساخت. این اکتیواکس ماشینهای زیادی را خاموش کرد ولی هیچ آسیبی به آنها نرساند و بلافاصله می‌شد ماشین را از نو راه‌اندازی نمود. واکنش رسمی به کار او آن بود که گواهینامه‌اش را برای این کنترل اکتیواکس خاص باطل کردند و بدین نحو به این داستان خجالت‌آور پایان داده شد ولیکن هنوز زمینهٔ بروز چنین مشکلاتی برای سوء استفادهٔ برنامه‌نویسان شرور وجود دارد. (Garfinkel with Spafford,2002) از آنجایی که هیچگاه نمی‌توان بر هزاران شرکت تولید نـرم‌افـزار کـه ممکن است کدهای متحرک بنویسند نظارت کرد، لذا روش امضای کدهای اجرایی، فاجعه‌ای است که هر لحظه در شرف وقوع خواهد بود.

اسکریپتهای جاوا

اسکریپتهای جاوا ذاتاً دارای هیچ مدل امنیتی رسمی و شناخته شده‌ای نیستند و پیاده‌سازیهای ناامن و نفوذپذیر آنها پیشینه‌ای طولانی دارد. هر تولیدکنندهٔ نرم‌افزار از یک دیدگاه خاص به امنیت می‌پردازد. به‌عنوان مثال نسخهٔ دوم از مرورگر نت‌اسکیپ از مدلی شبیه به مدل جاوا استفاده می‌کند (یعنی مبتنی بر نظارت و کنترل فراخوانیها)، در حالی که در نسخهٔ چهارم به سمت مدل امضای کدها حرکت کرده است.

مشکل اساسی آنست که اجازهٔ اجرای کدهای بیگانه و ناشناخته بر روی ماشین شما، نوعی خوشامدگویی به دردسر و گرفتاری است. از دیدگاه امنیت، این کار همانند آن است که یک دزد را به خانهٔ خود دعوت کرده و سپس تلاش کنید او را به دقت تحت نظر بگیرید مبادا از آشپزخانه به اتاق نشیمن فرار کند. اگر اتفاق غیرمنتظره‌ای بیفتد یا آن که برای لحظاتی غفلت کنید، حوادث ناخوشایندی می‌تواند رخ بدهد. گرایش به کدهای اجرایی و متحرک مثل اسکریپتها از آنجایی ناشی می‌شود که این کدها امکان نمایش گرافیکهای متحرک و تعاملات سریعتر و بهینه‌تری را فراهم می‌کنند و بسیاری از طراحان وب‌سایت وب فکر می‌کنند وجود این امکانات از امنیت مهمتر است بالاخص وقتی ماشین دیگران در معرض خطر باشد!!

ویروسها

ویروسها نوع دیگری از کدهای متحرک هستند. برخلاف مثالهای فوق، هیچکس ویـروسها را بـه مـاشین خـود دعوت و منتقل نمی‌کند. تفاوت بین یک ویروس و کدهای متحرک معمولی آن است که ویروسها به گونه‌ای نوشته می‌شوند که خود را تکثیر کنند. وقتی یک ویروس از طـریق صـفحات وب ، ضـمیمه‌های نـامهٔ الکترونیکی یـا روشهای دیگر به یک ماشین می‌رسد، کار خود را با آلوده کردن برنامه‌های اجرایی روی دیسک آغاز می‌کند، وقتی یکی از این برنامه‌ها اجرا شود، کنترل اجرا به ویروس منتقل شده و آن ویروس مجدداً تلاش می‌کند به روشهایی مثل ارسال کپی خودش از طریق پست الکترونیکی (به آدرسهایی که قربانی در کامپیوترش یادداشت کرده) ، خود را تکثیر کند. برخی از ویروسها بوت‌سکتور دیسک سخت را آلوده می‌کنند به نحوی که وقتی ماشین بوت می‌شود، ویروس نیز بلافاصله اجرا می‌گردد. ویروسها به یک مشکل عمده برای شبکهٔ اینترنت تبدیل شده‌اند و میلیاردها دلار خسارت به بار می‌آورند. شاید نسل جدید سیستمهای عامل که براساس تکنولوژی **ریزهستهٔ امن** (Secure Microkernel) و تفکیک دقیق و محکم کاربران ، پروسه‌ها و منابع بنا نهاده شده است به کاهش این مشکل کمک کند.

۸ـ۱۰ زمینه‌ها و پی‌آمدهای اجتماعی

اینترنت و تکنولوژی امنیت آن، موضوعی است که در آن موارد اجتماعی، سیاستهای عمومی و تکنولوژی با یکدیگر تلاقی می‌کنند و اغلب تبعات گسترده و عظیمی دارند. در زیر به اختصار سه موضوع را بررسی خواهیم کرد: «**حریم خصوصی افراد**» (Privacy)، «**آزادی بیان**» و «**حقوق مالکیت معنوی**» (Copy right). بدیهی است که فقط می‌توانیم این موارد را به صورت سطحی بررسی نماییم. برای مطالعه بیشتر در این خصوص به این مراجع مراجعه کنید: (Anderson, 2001; Garfinkel with Spafford, 2002; Schneier, 2000) در خصوص این موضوعات اینترنت سرشار از مطلب و مقاله است. فقط کافی است کلمات "Privacy" یا "Censonrship" (سانسور) یا "Copyright" را در یک موتور جستجو ، تایپ کنید و نتایج جستجو را دنبال نمایید. همچنین می‌توانید به وب‌سایت همین کتاب مراجعه کنید تا در این خصوص، چندین لینک به سایتهای مفید بدست بیاورید.

۸ـ۱۰ـ۱ حریم خصوصی افراد (Privacy)

آیا افراد از حقّ داشتن حریم خصوصی بهره‌مند هستند؟ سؤال بسیار خوبی است! در چهارمین اصلاحیهٔ قانون اساسی در ایالات متحدهٔ آمریکا آمده که حکومت بدون دلیل موجه و قانونی حق جستجوی منازل مردم، نوشته‌ها و آثار آنها را ندارد و شرایط صدور چنین مجوزی بسیار محدود و خاص در نظر گرفته شده است. لذا حداقل در ایالات متحده، بیش از ۲۰۰ سال است که حفظ حریم خصوصی افراد به یک قاعدهٔ عمومی تبدیل شده است.

در طول چند دههٔ گذشته چیزهایی تغییر کرده‌اند که هم کار حکومت در جاسوسی از شهروندان را ساده‌تر کرده و هم این امکان را برای شهروندان فراهم آورده که جلوی این جاسوسی را بگیرند! در قرن هجدهم، برای آن که دولت بتواند نوشته‌های یک نفر را تفتیش کند، مجبور بود که یک پلیس اسب سوار را به مزرعهٔ آن شهروند بفرستد و مستندات خاصی را بررسی نماید. این روال بسیار پر دردسر بود. امروزه شرکتهای تلفن و ارائه کنندگان خدمات اینترنت در صورت ارائهٔ مجوز لازم به سادگی امکان ایجاد انشعاب و استراق سمع را در اختیار می‌گذارند. این امکان، کار پلیسها را راحتتر کرده و خطر سقوط از اسب نیز وجود ندارد!!

رمزنگاری وضعیت را تغییر داده است. هر کسی که یک نسخه از بستهٔ نرم‌افزاری PGP را دریافت و نصب کند و از کلید رمز مطمئن و مستحکم استفاده نماید می‌تواند اطمینان داشته باشد که هیچکس در این جهان قادر به خواندن نامه‌های او نخواهد بود. دولتها و حکومتها از این موضوع به خوبی آگاهند و هرگز خشنود نیستند. حفظ حریم خصوصی افراد به صورت واقعی، بدین معناست که جاسوسی افراد در موارد جنایی نیز بسیار دشوار خواهد بود. همچنین جاسوسی روزنامه‌نگاران و رقبای سیاسی بسیار سخت‌تر می‌شود. بدین دلیل برخی از دولتها بکارگیری یا صدور محصولات رمزنگاری را محدود یا حتی ممنوع کرده‌اند. به عنوان مثال در فرانسه تا قبل از سال ۱۹۹۹ هرگونه رمزنگاری اطلاعات ممنوع بود مگر آن که کلید رمز به دولت تسلیم شده باشد.

فرانسه تنها نبود. در آوریل ۱۹۹۳ دولت آمریکا اعلام کرد که در نظر دارد یک سخت‌افزار رمزنگار به نام Cliper Chip را به عنوان استانداردی برای تمام مخابرات شبکه بسازد. البته عنوان شده بود که حریم خصوصی شهروندان محترم شمرده می‌شود. همچنین اشاره شده بود که این تراشهٔ عرضه شده توسط دولت قادر است تمام ترافیک داده‌ها را با استفاده از روشی به نام **Key escrow** رمزگشایی کند؛ این ساختار اجازه می‌داد تا دولت تمام کلیدهای رمز را در اختیار داشته باشد. با این وجود دولت قول داده بود که فقط با مجوز قانونی به جستجو و استراق سمع داده‌ها خواهد پرداخت. بدیهی است که خشم عموم برانگیخته شد، طرفداران حفظ حریم خصوصی شهروندان آن را شرم‌آور و ننگین دانستند در حالی که مجریان قانون آن را ستایش می‌کردند. عاقبت، دولت

عقب‌نشینی کرد و این نظریه را کنار گذاشت.

حجم بسیار زیادی اطلاعات مفید، در خصوص حریم الکترونیکی افراد در سایت متعلق به سازمان Electronic Frontier Foundation به آدرس *www.eff.org* در دسترس عموم قرار دارد.

نامه‌پراکن‌های ناشناس (Anonymous Remailer)

PGP، SSL و تکنولوژیهای دیگر این امکان را فراهم آورده‌اند که دو طرف با یکدیگر به صورت امن و احراز هویت شده و فارغ از شنود یا دخالت شخص ثالث مبادله و مخابرهٔ اطلاعات داشته باشند. ولیکن گاهی حریم خصوصی افراد ایجاب می‌کند که احراز هویت وجود نداشته باشد یعنی افراد بتوانند به صورت ناشناس با یکدیگر در ارتباط باشند.

اجازه بدهید به چند مثال بپردازیم: اول آن که مخالفان سیاسی که در سیطرهٔ رژیمهای استبدادی زندگی می‌کنند اغلب علاقه‌مندند به صورت ناشناس ارتباط برقرار کنند تا از دستگیری و نهایتاً کشته شدن در امان بمانند. دوم آن که خلافکاری شرکتها، مؤسسات آموزشی، دولتی و دیگر سازمانها اغلب توسط افرادی فاش شده که می‌خواهند برای فرار از انتقام ناشناس بمانند. سوم، افرادی که دارای دیدگاههای اجتماعی، سیاسی یا مذهبی نامتداول یا مخالف هستند علاقه‌مندند از طریق پست الکترونیکی یا گروههای خبری بدون افشای هویت خود با یکدیگر ارتباط برقرار کنند. چهارم، شاید افرادی بخواهند مسائل خود در خصوص بیماریهای روانی و مواردی از این قبیل را به صورت ناشناس در گروههای خبری (newsgroup) بیان کنند. البته مثالهای بی‌شمار دیگری نیز وجود دارد.

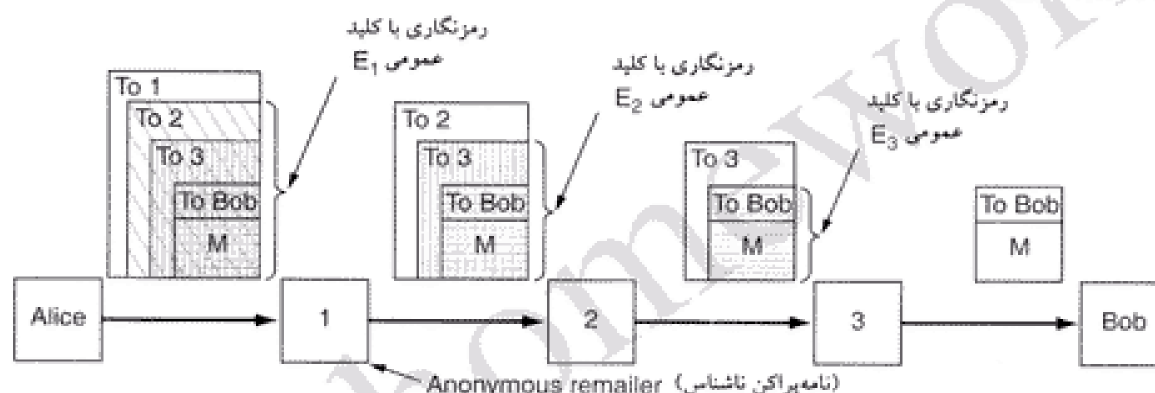
بیائید به یک مثال خاص بپردازیم. در سال ۱۹۹۰ برخی از مخالفین یک اقلیت مذهبی، دیدگاههای خود را از طریق یک سیستم نامه‌پراکن ناشناس (Anonymous Remailer) برای یک گروه خبری در یوزنت ارسال کردند. این سرویس‌دهنده به کاربران اجازه می‌داد تا برای خود اسم مستعار انتخاب کرده و نامه‌های الکترونیکی خود را برای آن بفرستند؛ آن سرویس دهنده نیز نامه‌ها را با اسم مستعار برای علاقمندان ارسال می‌کرد؛ بدین ترتیب هیچکس نمی‌توانست بگوید که پیام از طرف چه کسی آمده است. در برخی از این مرسولات اطلاعاتی فاش شده بود که این اقلیت مذهبی بعداً ادعا کرد اسرار بازرگانی و اسناد دارای حق مالکیت (Copyright) آنها بوده است. سران این اقلیت مذهبی با گزارش به مراجع قانونی ادعا کردند که اسرار بازرگانی آنها فاش و مالکیت معنوی اسناد نقض شده است و هر دوی این موارد در محلی که این سرویس دهنده قرار داشت جرم محسوب می‌شد. مورد به دادگاه کشیده شد و اپراتور این سرویس دهنده وادار شد اطلاعات مربوط به فهرست اصلی افراد و اسامی مستعار را به دادگاه عرضه کند و بدین ترتیب هویت واقعی افرادی که با این سرویس‌دهنده مکاتبه‌ای داشتند مشخص شد. (این اولین باری نبود که یک گروه مذهبی از انتشار اسرار درونی خود بر می‌آشفت: ویلیام تیندال در سال ۱۵۳۹ به دلیل ترجمهٔ انجیل به زبان انگلیسی در آتش خشم سوخت.)

بخش قابل ملاحظه‌ای از جامعهٔ اینترنت از بابت نقض حریم خصوصی و افشای اسرار دیگران در ماجرای فوق به خشم آمد. نتیجه‌ای که عموم افراد از این ماجرا گرفتند آن بود که یک سرویس‌دهندهٔ نامه‌پراکن ناشناس که در آن آدرسهای واقعی نامه‌های الکترونیکی و اسامی مستعار ذخیره می‌گردد (که سیستم **نامه‌پراکن نوع یک** Type 1 Remailer نامیده می‌شود) ارزش و قابلیت اعتماد چندانی ندارد. این موارد بسیاری از افراد را بر آن داشت تا سیستمهای نامه‌پراکن ناشناس را به‌گونه‌ای طراحی کنند که بتواند در مقابل حملاتی نظیر احضار و توقیف سرویس دهنده مقاومت کند.

این سیستمهای جدید نامه‌پراکن که اغلب *Cypherpunk Remailer* نامیده می‌شوند عملکردی شبیه به روال زیر دارند: کاربر پیام الکترونیکی خود را تولید و سرآیند استاندارد RFC 822 را بدان می‌افزاید (البته بدون گزینهٔ :From) و سپس آن را با کلید عمومی سرویس‌دهندهٔ نامه‌پراکن رمز می‌کند و نهایتاً پیام را برای آن نامه‌پراکن

می‌فرستد. در این سرویس دهنده، سرآیند خارجی RFC 822 حذف و پیام رمزگشایی می‌شود و سپس برای دیگران ارسال می‌گردد. سرویس دهندهٔ نامه‌پراکن، هیچگونه حساب کاربری (Account) تعریف نکرده و هیچ **«فایل سوابق»** (Log File) ذخیره و نگهداری نمی‌کند و بدین ترتیب حتی اگر بعداً این سرویس‌دهنده توقیف شود هیچ ردّپایی از پیامهایی که از آن گذر کرده‌اند بجا نمی‌ماند.

بسیاری از کاربران که علاقه‌مند و مصّر به ناشناس ماندن هستند به نحوی که در شکل ۸-۵۴ نشان داده شده، پیامهای خود را به صورت زنجیره‌ای از چندین نامه‌پراکن عبور می‌دهند. در این شکل، آلیس تمایل دارد که یک پیام را به صورت کاملاً ناشناس برای باب بفرستد. به همین دلیل از سه سرویس دهندهٔ نامه‌پراکن استفاده می‌کند. او پیام M را تنظیم کرده و سرآیند لازم را که آدرس پست الکترونیکی باب جزو آنست به پیام اضافه می‌کند. سپس کل آن را با کلید عمومی سومین سرویس‌دهندهٔ نامه‌پراکن یعنی E3 رمز می‌کند. (در شکل این پیام به صورت هاشورهای افقی نشان داده شده است.) سپس به پیام سرآیند جدیدی که در برگیرندهٔ آدرس پست الکترونیکی سرویس دهندهٔ سوم است اضافه می‌کند. این پیام در حقیقت همانی است که در شکل، بین سرویس دهنده ۲ و ۳ مبادله شده است.



شکل ۸-۵۴. چگونگی ارسال پیام ناشناس از آلیس به باب به کمک سه نامه‌پراکن ناشناس.

سپس پیام جدید را با کلید عمومی سرویس‌دهندهٔ نامه‌پراکن دوم رمز می‌کند. (در شکل این پیام با هاشورهای عمودی نشان داده شده است.) در ادامه سرآیندی حاوی آدرس پست الکترونیکی سرویس‌دهندهٔ دوم به متن حاصل می‌افزاید. این پیام در شکل ۸-۵۴ بین سرویس دهندهٔ ۱ و ۲ نشان داده شده است. نهایتاً او کل پیام را با کلید عمومی سرویس دهندهٔ نامه‌پراکن ۱ رمز کرده و آدرس پست الکترونیکی او را به آن اضافه می‌کند. این پیام همانی است که بین آلیس و سرویس دهندهٔ اول در شکل مبادله می‌شود و پیام واقعی که انتقال داده خواهد شد همین پیام است.

وقتی پیام به اولین سرویس‌دهندهٔ نامه پراکن بر می‌خورد، سرآیند بیرونی آن جدا شده و متن درون آن رمزگشایی و برای سرویس دهندهٔ دوم ارسال می‌شود. همین مراحل در دو سرویس‌دهندهٔ بعدی اتفاق می‌افتد.

اگرچه تعقیب ردّ پیامها برای هر کس بی‌نهایت دشوار است ولیکن برای اطمینان بیشتر، بسیاری از این نامه‌پراکنها اقدامات احتیاطی اضافه‌تری را انجام می‌دهند. به عنوان مثال ممکن است پیام را به صورت تصادفی برای مدتی معطل کنند یا اطلاعات زائدی را به انتهای پیام بچسبانند یا آنها را حذف کنند یا ترتیب پیامها را عوض نمایند تا هیچکس براحتی نتواند تشخیص بدهد کدام پیام خروجی متناظر با کدام پیام ورودی است و به کدام سرویس‌دهندهٔ نامه‌پراکن می‌رود و بدین ترتیب حمله‌ای که براساس تحلیل ترافیک صورت می‌گیرد خنثی خواهد شد. برای شرح بیشتر در خصوص سیستم پست الکترونیکی ناشناس و مدرن به مراجع (Mazieres and Kaashoek, 1998) مراجعه نمائید.

ناشناس ماندن (Anonimity) فقط محدود به نامه‌های الکترونیکی نیست. برای گشت و گذار ناشناس در وب نیز سرویسهایی وجود دارد. کاربر، مرورگر خود را به گونه‌ای تنظیم می‌کند تا از سرویس‌دهندهٔ Anonymizer به عنوان **پراکسی** استفاده نماید. از آن پس تمام تقاضاهای HTTP به سوی این سرویس‌دهنده ارسال می‌شود و این سرویس دهنده به نیابت از کاربر صفحات را تحویل گرفته و به او بر می‌گرداند. تمام وب‌سایتهای دنیا، این سرویس‌دهنده را (یعنی Anonymizer را) مبداء اصلی تقاضا می‌بینند نه کاربر اصلی را. تا زمانی که سرویس‌دهندهٔ Anonymizer از نگهداری «فایل سوابق» (Log) اجتناب ورزد هیچکس نمی‌تواند تعیین کند چه کسی تقاضای کدام صفحه را داده است.

۸-۱۰-۲ آزادی بیان

رعایت حریم خصوصی (Privacy) خواسته افرادی است که تمایل دارند آنچه دیگران می‌توانند در ارتباط با آنها ببینند و بدانند محدود باشد. یکی دیگر از موارد مهم اجتماعی، «آزادی بیان» و متضاد آن «سانسور عقاید» است. حکومتها می‌خواهند آنچه را که افراد می‌توانند بخوانند یا منتشر کنند، محدود باشد. وب که حاوی میلیونها میلیون صفحهٔ حاوی اطلاعات است به بهشت سانسورگران تبدیل شده است. بسته به ماهیت و ایدئولوژی یک ملت، مفاد ممنوعه در وب می‌تواند شامل موارد ذیل باشد:

۱. مفادی که برای کودکان و نوجوانان مناسب نیست.
۲. تنازعات قومی، نژادی، مذهبی و گروهی
۳. اطلاعاتی شامل آموزش جرم و جنایت
۴. اسرار ساخت جنگ‌افزارهای کشتار جمعی
۵. تبلیغ مسائل ضدّملی و ضدّاجتماعی

واکنش معقول آن است که سایتهای نامناسب باید ممنوع و توسعه‌دهندگان آن تحت پیگرد قرار بگیرند.

ولیکن برخی از بایدها و نبایدها در تعارض با یکدیگر قرار می‌گیرند و تفکر همهٔ افراد، ملتها و حکومتها یکی نیست. به عنوان مثال در نوامبر ۲۰۰۰ دادگاهی در فرانسه به سایت یاهو مستقر در فرانسه اخطار داد که اجازه ندهد شهروندان فرانسوی سایت حراج خاطرات نازیها را ببینند زیرا در اختیار داشتن چنین مواردی طبق قانون فرانسه جرم محسوب می‌شود. چنین مواردی بسیار فراوان است. هر کشور قوانین و قواعد خاص خود را دارد و قوانین آن با آنچه که در وب جریان دارد همسو نیست. در طرف مقابل گسترهٔ وب جهانی است و نمی‌توان هیچ قانون واحدی را بر آن حاکم کرد. از طرفی امروزه سرویسهایی عرضه شده که اگرچه در مقابل پدیدهٔ سانسور قرار می‌گیرند ولیکن می‌تواند درهم شکنندهٔ حریم اخلاقی، اجتماعی یا ملی کشورها محسوب شود. نوعی از این سرویس که اصطلاحاً «سرویس ازلی یا Eternity Service» نام دارد در ابتدا به عنوان سیستمی مقاوم در برابر سانسور پیشنهاد شد. (Anderson,1996) بعداً سیستمهای کاملتر دیگری پیشنهاد و بعضاً پیاده‌سازی شدند. به این سرویس‌دهنده‌ها امکاناتی نظیر رمزنگاری، ناشناس ماندن افراد و تحمل خطا و خرابی (Fault Tolerance) اضافه شده است. فایلهایی که باید ذخیره شوند به قطعاتی تقسیم شده و این قطعات بر روی سرویس‌دهنده‌های متعدد ذخیره می‌گردد. در این سیستمها برخی از فایلها تا مدت زمان خاصی حتی توسط صاحب آن قابل تغییر و حذف نیستند و چون دارای پشتیبانهای متعدد (Backup) هستند حتی در صورت از کار افتادن یا توقیف یکی از آنها، سرویس دهنده‌های دیگر، عرضهٔ اطلاعات را ادامه خواهند داد. برخی از سیستمهای پیشنهادی در این خصوص عبارتند از: Freenet (Clark, 2002)، PASIS (Wylie, 2000) و Publius (Waldman, 2000). موارد دیگر در مرجع (Serjantov, 2002) گزارش شده‌اند.

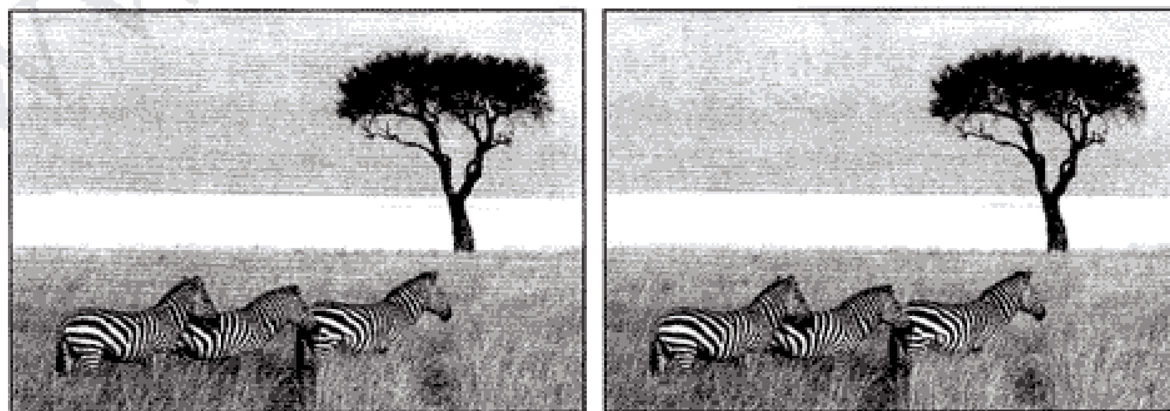
بسیاری از کشورها به صورت فزاینده‌ای در تلاش هستند تا صدور محصولات غیرعینی و غیرفیزیکی را که بیشتر در برگیرندهٔ سایتهای وب، نرم‌افزار، مقالات علمی، نامه‌های الکترونیکی و مشاوره‌های تلفنی هستند، قانونمند و محدود کنند. حتی انگلستان که برای قرنها مدعی آزادی بیان بوده، بطور جدی در حال بررسی قوانین محدودکننده و سرسختانه‌ای است که در آن به عنوان مثال مباحثات بین یک استاد بریتانیایی و دانشجوی خارجی او در دانشگاه کمبریج در شمول این قانون قرار گرفته و نیاز به مجوز از دولت دارد. (Anderson, 2002) بدیهی است که چنین سیاستهایی مناقشه برانگیز هستند.

استیگانوگرافی [1] (پوشیده‌نویسی)

در کشورهایی که حجم سانسور بسیار زیاد است مخالفان اغلب سعی می‌کنند برای فرار از آن از تکنولوژی استفاده نمایند. رمزنگاری اجازه می‌دهد که پیامها (حتی اگر قانونی هم نباشند) ارسال شوند ولیکن اگر حکومت، آلیس را شخصی بد و مخرّب بینگارد، افرادی مثل باب که با او مراوده دارند را نیز همفکر او تلقی می‌کند. سرویس‌دهندهٔ «نامه‌پراکن ناشناس» (Anonymous Remailer) می‌تواند به آنها کمک کند ولی اگر چنین سرویس‌دهنده‌هایی تحریم و مسدود شده باشند و یا ارسال پیام به یکی از آنها در خارج، نیاز به مجوز دولت داشته باشد، چندان مفید نخواهند بود؛ ولی وب راه‌گشا است.

افرادی که می‌خواهند به صورت سرّی با یکدیگر ارتباط داشته باشند اغلب سعی می‌کنند این ارتباط را به هر نحوی پنهان نگاه دارند. علم مخفی کردن پیامها اصطلاحاً «**استیگانوگرافی**» نامیده می‌شود که برگرفته از دو کلمهٔ یونانی به معنای «پوشیده‌نویسی» است. در حقیقت در ابتدا یونانیان باستان خود از این روش استفاده کرده‌اند. «هرودوت» مورخ یونانی از یک ژنرال ارتش یاد کرده که سر پیام‌رسان خود را تراشید و پیام را بر روی پوست سر او خالکوبی کرد؛ سپس صبر کرد تا قبل از اعزام او به مأموریت، موهای او رشد کنند! تکنیکهای مدرن از لحاظ مفهوم مشابه همین روش هستند!

به عنوان یک مثال از استیگانوگرافی به شکل ۸–۵۵–الف دقت کنید. این عکس توسط مؤلف کتاب در کنیا گرفته شده است و در آن سه گورخر در کنار درخت اقاقیا دیده می‌شوند. تصویر ۸–۵۵–ب نیز مشابه با عکس قبلی به نظر می‌رسد ولیکن این عکس جذابیتهای دیگری هم دارد! این عکس حامل متن کامل پنج نمایشنامه از شکسپیر است که مخفیانه درون آن جاسازی شده است، شامل: «هملت»، «شاه‌لیر»، «مکبث»، «تاجر ونیزی» و «ژولیوس سزار». مجموع این نمایشنامه‌ها جمعاً ۷۰۰ کیلوبایت متن هستند.



(الف) (ب)

شکل ۸–۵۵. (الف) سه گورخر و یک درخت! (ب) سه گورخر و یک درخت و متن کامل ۵ نمایشنامه از شکسپیر!!

---

۱. Steganography

این مخفی‌سازی چگونه انجام می‌شود؟ ابعاد تصویر رنگی و اصلی ۷۶۸×۱۰۲۴ نقطه (پیکسل) است. هر پیکسل شامل سه عدد هشت بیتی است که هر یک شدت رنگهای قرمز، سبز و آبی را در هر نقطهٔ تصویر مشخص می‌کنند. از ترکیب این سه رنگ (با شدتهای متفاوت) رنگ هر نقطه بدست می‌آید. در روش کدگذاری مخفی از کم‌ارزش‌ترین بیت هر یک از سه مقدار رنگهای RGB به عنوان «کانالهای مخفی» (Covert Channel) استفاده می‌شود. بنابراین هر پیکسل فضایی معادل ۳ بیت برای جاسازی اطلاعات سرّی در اختیار می‌گذارد؛ (یک بیت در مقدار قرمز، یکی در آبی و یکی در سبز). در تصویری به ابعاد فوق مجموعاً ۳×۷۶۸×۱۰۲۴ بیت (معادل ۲۹۴۹۱۲ بایت) از اطلاعات سرّی را می‌توان جاسازی کرد.

متن کامل این پنج نمایشنامه به همراه یک توضیح کوتاه جمعاً حدود ۷۳۴۸۹۱ بایت است. این متن ابتدا با استفاده از الگوریتم استاندارد فشرده‌سازی، به ۲۷۴ کیلوبایت فشرده شده و سپس نتیجه ، با استفاده از الگوریتم IDEA رمزنگاری و در کم ارزش‌ترین بیت از مقادیر رنگها ذخیره شده است. به گونه‌ای که مشاهده می‌شود (یا به عبارت بهتر به گونه‌ای که مشاهده نمی‌شود!!) وجود این اطلاعات کاملاً غیرقابل رویت است. حتی در تصویر بزرگ شده و تمام رنگی این عکس باز هم چیزی قابل رویت نیست. چشم نمی‌تواند براحتی تفاوت بین رنگهای ۲۱ بیتی یا ۲۴ بیتی را تشخیص بدهد.

البته مشاهدهٔ دو تصویر فوق به صورت سیاه و سفید و با دقت پایین در این کتاب، نمی‌تواند در مورد قدرت این تکنیک قضاوت کند. برای آن که عملکرد استیگانوگرافی را بهتر احساس کنید، مؤلف یک نمونهٔ نمایشی از استیگانوگرافی، شامل تصویر تمام رنگی با دقت بالا از شکل ۸-۵۵-ب را به همراه پنج نمایشنامهٔ جاسازی شده، عرضه کرده است. این نمونه نمایشی شامل یک ابزار برای جاسازی و استخراج متن در تصویر است که می‌توانید آن را در وب‌سایت این کتاب بدست بیاورید.

برای استفاده از استیگانوگرافی به منظور محاورهٔ مخفیانه، معاندین می‌توانند یک وب‌سایت ایجاد کنند که سرشار از تصاویر مُجاز و قانونی باشد. در حالی که پیامهای مخفی در آن جاسازی شده است. اگر پیامها ابتدا فشرده و سپس رمزنگاری شوند حتی در صورتی که کسی به وجود پیام مخفی در آن شک کند قطعاً برای او تشخیص پیام از نویز سفید تصویر، بسیار دشوار خواهد بود.

تصاویر بهیچ‌وجه تنها حامل پیامهای مخفی نیستند. فایلهای صوتی نیز بخوبی کارآیی دارند. فایلهای ویدیویی دارای پهنای باند بسیار عظیمی برای پنهان‌سازی اطلاعات هستند. حتی ترکیب چیده شدن عناصر (Layout) و ترتیب برچسبها در فایل HTML (HTML Tags) نیز می‌تواند حامل اطلاعات باشد!

استیگانوگرافی تنها برای حمل اطلاعات مخفی نیست و کاربردهای دیگری هم دارد. یکی از کاربردهای عمومی آن می‌تواند این باشد که صاحب حقوقی یک عکس پیامهائی سرّی در درون یک تصویر جاسازی کند. هر گاه چنین تصویری دزدیده شده و در یک وب‌سایت قرار داده شود، مالک قانونی آن می‌تواند این پیام محرمانه و سرّی را برای اثبات مالکیت آن، به دادگاه عرضه کند. به این تکنیک اصطلاحاً نشانه‌گذاری (Watermarking) گفته می‌شود و در مرجع (Piva et al., 2002) تشریح شده است.

برای بدست آوردن اطلاعات بیشتر در خصوص استیگانوگرافی به مراجع ذیل مراجعه کنید:

(Artz, 2001; Johnson and Jajoda, 1998; Katzenbeisser and Petiteolas, 2000; and Wayner, 2002)

۸-۱۰-۳ مالکیت معنوی (Copyright)

رعایت حریم خصوصی افراد و پدیدهٔ سانسور موضوعاتی هستند که سیاستهای عمومی و تکنولوژی را رو در رو قرار داده‌اند. مورد سوّم **مالکیت معنوی آثار** است. مالکیت معنوی (Copyright) بدین معناست که امتیاز

بهره‌برداری از عواید یک اثر برای مدتی برای آفرینندگان آن (شامل نویسندگان، هنرمندان، آهنگسازان، نوازندگان، عکاسان، سینماگران) محفوظ بماند که این دورهٔ زمانی می‌تواند ۵۰ تا ۷۵ سال پس از عمر صاحب اثر باشد. پس از آن که طول دورهٔ این امتیاز که به صورت قانونی ثبت می‌شود به سر آمد، آن اثر همگانی تلقی شده و همه می‌توانند به دلخواه از آن بهره ببرند یا آن را بفروشند. به عنوان مثال پروژهٔ گوتنبرگ یا آثار شکسپیر امروزه همگانی تلقی می‌شوند و به رایگان در وب در دسترس هستند. در سال ۱۹۹۸ کنگرهٔ آمریکا طول زمان مالکیت آثار را به درخواست هالیوود که مدعی شده بود اگر افزایش نیابد هیچکس قادر نخواهد بود چیزی خلق کند، ۲۰ سال اضافه کرد.

مناقشه بر سر موضوع مالکیت آثار، زمانی اوج گرفت که تعداد مشترکین شرکت Napster (ارائه دهندهٔ خدمات مبادلهٔ رایگان موزیک) به ۵۰ میلیون نفر رسید. از آنجایی که Napster هیچگونه عمل کپی فایلهای موزیک را انجام نمی‌داد دادگاه به طرح این اتهام پرداخت که در اختیار گذاشتن یک بانک اطلاعاتی از اینکه چه کسی چه موزیکهایی را در اختیار دارد معاونت در جرم تلقی می‌شود و Napster بدین ترتیب به وقوع جرم کمک کرده است. اگرچه هیچکس مخالف قانون مالکیت معنوی آثار نیست (هرچند برخی ادعا می‌کنند طول دوره و جزئیات سختگیرانهٔ آن زیاد است) ولیکن دور جدیدی از به اشتراک‌گذاری موزیک، یک مناقشه عظیم را دامن زده است.

به عنوان مثال شبکه‌ای نقطه‌به‌نقطه را در نظر بگیرید که در آن افراد فایلهای قانونی خود را (شامل موزیکهای همگانی، ویدیوهای شخصی، اعلانهای مذهبی که در آن اسراری وجود ندارد) و شاید چند فایل که حق مالکیت آن انحصاری است را به اشتراک گذاشته‌اند. فرض کنید افراد به صورت دائم از طریق یک خط ADSL یا کابل، به شبکه وصل شده‌اند. هر ماشین یک فهرست از آنچه بر روی دیسک سخت او موجود است و همچنین فهرستی از بقیهٔ مشترکین را در اختیار دارد. هر کسی که به دنبال یک مورد خاص می‌گردد یکی از اعضای این فهرست را انتخاب کرده و بررسی می‌کند که آیا او این آیتم را در اختیار دارد یا خیر؟ اگر نداشت او می‌تواند در فهرستِ یکایک اعضا بررسی کند. پس از پیدا شدن آیتم مورد نظر، متقاضی می‌تواند آن را کپی نماید.

اگر آنچه که به اشتراک گذاشته می‌شود تحت حمایت قانون مالکیت معنوی باشد، آنهایی که چنین آیتمهایی را برداشت می‌کنند مرتکب قانون‌شکنی شده‌اند. (هر چند مشخص نیست وقتی که انتقال چنین آیتمهایی به صورت بین‌المللی انجام می‌شود، چه قانونی قابل‌اعمال و استناد خواهد بود.) با کسی که چنین زمینه‌ای را فراهم کرده چه باید کرد؟ آیا این جرم است که یک موزیک را که شما بهای آن را پرداخته و بر روی دیسک سخت خود ذخیره کرده‌اید، دیگران پیدا و دریافت کنند؟ اگر مثلاً شما درب اتاق خود را قفل نکرده باشید و یک دزد بتواند از روی یکی از کتابهای شما کپی بگیرد، آیا شما در جرم نقض حقوق قانونی آن کتاب، گناهکار هستید؟

مناقشات بسیار گسترده‌ای پیرامون مالکیت معنوی آثار در گرفته است و نزاع شدیدی بین هالیوود و شرکتهای کامپیوتری بوجود آمده است. هالیوود می‌خواهد تا قوانین حفاظت از آثار، سختگیرانه و سنگین‌تر شود و شرکتهای کامپیوتری نیز نمی‌خواهند مأمور حفاظت از منافع هالیوود باشند.

در اکتبر ۱۹۹۸، کنگرهٔ آمریکا قانونی به نام DMCA (Copyright Act Digital Millennium) را از تصویب گذراند که براساس آن هر گونه فریبکاری و حیله‌پردازی برای نقض مالکیت آثار در پوششهای به ظاهر قانونی یا انتشار چنین راههایی برای فرار از قانون، جرم محسوب می‌شود. چنین قانونی در اروپای متحد نیز از تصویب گذشت. در حالی که بسیاری از افراد آگاه نیستند که در شرق دور نسخه‌برداری از آثار مجاز شمرده می‌شود! و خوشبینانه بدان می‌اندیشند که قانون DMCA می‌تواند بین حقوق مالکین و پدیدآورندگان آثار و حقوق عمومی یک توازن ایجاد کند.

اشاره به موردی دیگر خالی از لطف نیست. در سپتامبر ۲۰۰۰، یک کنسرسیوم از صنایعی که در موزیک فعالیت می‌کردند سیستمی غیرقابل‌نفوذ برای فروش موزیک (به صورت On-line) را سازماندهی و ایجاد کرده و سپس از رقبا خواستند تا افراد را به شکستن این سیستم دعوت نمایند (که عملی کاملاً قانونی برای هر سیستم امنیتی جدید است چراکه به آشکار شدن اشکالات سیستم کمک می‌کند.) یک گروه از محققین امنیت سیستم از چندین دانشگاه به سرپرستی پروفسور ادوارد فِلتن از دانشگاه پرینستون بدین چالش علمی وارد شده و این سیستم را درهم شکستند. سپس مقاله‌ای در خصوص یافته‌های خود نوشته و آن را برای کنفرانس امنیت USENIX ارسال کردند. قبل از آن که موعد ارائه این مقاله فرا برسد، فِلتن نامه‌ای از انجمن صنایع ضبط و پخش موزیک در آمریکا دریافت کرد که او را تهدید کرده بودند در صورت انتشار مقاله، بر علیه او طبق قانون DMCA ادعای خسارت خواهند کرد.

در پاسخ، فلتن از دادگاه فدرال کسب تکلیف کرد که آیا تألیف مقالات علمی در خصوص امنیت سیستمها قانونی است یا خیر؟ این انجمن از ترس آن که دادگاه بر علیه آنها کاری انجام بدهد دست از تهدید فلتن برداشتند و پرونده مختومه شد. شکی نیست که این صنایع اشکال کار را در خودشان می‌دیدند: از یک طرف افراد را دعوت به شکستن سیستم کرده و از طرف دیگر آنهایی که چالش آنها را پذیرفته بودند تهدید به ادعای خسارت و شکایت کردند. پس از آن که تهدید رفع شد مقالهٔ فوق‌الذکر منتشر گردید.

مورد دیگری که به بحث ما مرتبط است بسط **«نظریهٔ استفادهٔ جوانمردانه از آثار»** (Fair Use Doctorine) است که توسط قانونگذاران قوهٔ قضاییه در بسیاری از کشورها وضع شده است. این نظریه بیان می‌کند که خریداران یک اثر که حقوق معنوی آن متعلق به دیگران است، اجازه دارند طبق ضوابط خاصی از آن کار نسخه‌برداری کنند یا بخشهایی از آن را در جهت مقاصد علمی نقل قول کنند، مفاد آن را در دانشگاه یا مدارس تدریس نمایند و حتی برای اطمینان خاطر از آنکه در صورتی خرابی نسخهٔ اصلی یک اثر چیزی از دست ندهند، از آن چندین نسخهٔ کپی تهیه کنند. برای بررسی آن که آیا از یک اثر، استفادهٔ جوانمردانه می‌شود یا نه، باید معیارهای زیر ارزیابی شود: (۱) آیا استفاده از آن اهداف تجاری دارد. (۲) چند درصد از کل آن نسخه‌برداری می‌شود. (۳) نسخه‌برداری از آن بر فروش آن اثر چقدر تأثیر منفی دارد. از آنجایی که قانون DMCA و قوانین مشابه در اروپا، هرگونه روشهای زیرکانه و فریبکارانه برای پایمال کردن حقوق معنوی آثار را غیرمجاز می‌داند، استفادهٔ جوانمردانه و طبیعی با معیارهای فوق الذکر را نیز غدغن کرده است.

زمانی که که قانون DMCA تلاش داشت بین حقوق قانونی پدیدآورندگان آثار و حقوق استفاده کنندگان توازن معقول ایجاد کند، طرح جدیدی به رهبری اینتل و مایکروسافت به نام TCPA[1] ارائه شد. نظریه آن بود که یک تراشهٔ CPU و سیستم عامل آن، بدقت بر رفتار و عملکرد کاربران نظارت داشته باشد (مثل اجرای یک موزیک یا نرم‌افزار که به صورت غیرقانونی کپی شده است) و جلوی اعمال غیرمجاز کاربر را بگیرد. این سیستم حتی به مالکان نرم‌افزارها یا دیگر کالاهای الکترونیکی اجازه می‌دهد تا در هر زمان که صلاح دیدند از راه دور به PC کاربران سرکشی کرده و قواعد استفاده از محصولاتشان را تغییر بدهند. بدیهی است که تبعات اجتماعی چنین طرحی، بسیار زیاد خواهد بود. اگرچه توجه صاحبان صنایع به موضوع امنیت پسندیده و قابل تحسین است ولی بسیار ناگوار است که آنها به جای پرداختن به مسئله ویروسها، کراکرها (Crackers)، اخلالگران و دیگر موارد امنیتی که مردم با آن دست به گریبانند، تمام تلاش خود را مصروف اجرای قانون حمایت از پدیدآورندگان آثار کرده‌اند!!

کوتاه سخن آن که قانونگذاران و وکلا در سالهای آتی نیز درگیر مسئله توازن بین حقوق مصرف‌کننده و حقوق

---

۱. Trusted Computing Platform Alliance

پدیدآورندگان آثار خواهند بود. دنیای الکترونیکی امروز بی‌شباهت به صحنهٔ جنگ نیست: گروهی به جان گروه دیگر می‌افتند، یکدیگر را لگدمال می‌کنند، کارشان به دادگاه می‌کشد و خوشبختانه سرانجام کار، اکثراً مصالحه می‌کنند و این روال ادامه خواهد یافت تا در آخر تکنولوژی جدید از راه برسد.

۱۱ـ۸ خلاصه

رمزنگاری ، ابزاری مؤثر برای محرمانه نگاه داشتن اطلاعات و اطمینان از صحت و هویت آنهاست. سیستمهای رمزنگاری جدید براساس قانون کِرکهُف بنا شده‌اند یعنی الگوریتم بکار رفته در آنها آشکار و عمومی و فقط کلید رمز سرّی است. بسیاری از الگوریتمهای رمزنگاری از تبدیلهای پیچیدهٔ ریاضی شامل عملیات جانشینی و جایگشتی برای تبدیل متن به رمز بهره گرفته‌اند. ولیکن هر گاه رمزنگاری کوانتومی بتواند در محیط عمل وارد شود، روش One-Time Pad یک سیستم رمزنگاری واقعاً غیرقابل شکست عرضه خواهد کرد.

الگوریتمهای رمزنگاری را می‌توان به دو دسته تقسیم کرد: الگوریتمهای با کلید متقارن و الگوریتمهای با کلید عمومی. الگوریتمهای با کلید متقارن، بیتهای متن را در چندین مرحله و براساس پارامترهای مشتق شده از کلید اصلی، ترکیب و مخلوط می‌کنند تا در نتیجه متن رمز شده بدست آید. در حال حاضر الگوریتمهای Triple DES و Rijndael (AES) مشهورترین الگوریتمهای با کلید متقارن هستند. از این الگوریتمها می‌توان در حالتهای Electronic Code Book ، Cipher Block ، Chaining Mode ، Stream Cipher Mode ، Counter Mode و نظایر آن بهره گرفت.

الگوریتمهای رمزنگاری با کلید عمومی این ویژگی را دارند که کلیدهای رمزنگاری و رمزگشایی متفاوت از یکدیگر هستند و نمی‌توان با داشتن کلید رمزنگاری، کلید رمزگشایی را استخراج کرد. این ویژگی اجازه می‌دهد تا بتوان کلیدهای عمومی را منتشر کرد. اصلی‌ترین الگوریتم کلید عمومی، RSA است که قدرت خود را از آنجایی بدست آورده که تجزیهٔ اعداد بزرگ به عوامل اول بسیار بسیار دشوار است.

اسناد قانونی، تجاری و نظایر آن نیازمند امضاء هستند. بر این اساس روشهای متنوعی برای امضای دیجیتالی ابداع شده که در آن، هم از الگوریتمهای رمزنگاری با کلید متقارن و هم از روشهای کلید عمومی بهره گرفته شده است. بطور معمول، ابتدا از پیامهایی که باید امضا شوند با استفاده از روشهایی مثل MD5 یا SHA-1 یک رشته Hash (رشتهٔ خلاصه و درهم شدهٔ پیام) استخراج شده و سپس این رشته به جای متن اصلی رمزنگاری می‌شود.

مدیریت کلیدهای عمومی با استفاده از گواهینامه‌های دیجیتالی که در آن هویت شخص و کلید عمومی او درج شده، قابل انجام است.گواهینامه‌های دیجیتالی توسط مراکز معتمد مردم یا اشخاص حقیقی تائید می‌شوند. «ریشه» (یعنی Root یا عالیترین مرکز گواهی امضاء) باید پیشاپیش مشخص باشد ولیکن اغلب مرورگرها گواهینامهٔ دیجیتالی بسیاری از مراکز اصلی گواهی امضاء را به صورت درونی در اختیار دارند.

ابزارهای رمزنگاری می‌توانند برای امن کردن ترافیک جاری بر روی شبکه به کار گرفته شوند. IPsec در سطح لایهٔ شبکه عمل می‌کند و بسته‌هایی را که از یک ماشین به ماشین دیگر روانه می‌شوند، رمزنگاری می‌کند. دیوارهای آتش می‌توانند بر ورود و خروج اطلاعات یک سازمان نظارت کنند که این کار اغلب براساس نوع پروتکل (شمارهٔ پروتکل لایهٔ انتقال) و شمارهٔ پورت انجام می‌گیرد. شبکه‌های VPN می‌توانند شبکه‌های قدیمی که مبتنی بر خطوط اجاره‌ای و انحصاری بودند را با سطح امنیت مورد نظر شبیه‌سازی کنند. نهایتاً شبکه‌های بی‌سیم نیاز به امنیت خوب و کافی دارند در حالی که شبکهٔ 802.11 WEP چنین امنیتی را فراهم نکرده است؛ اگرچه 802.11i خواهد توانست این ویژگی را بهبود بخشد.

وقتی دو طرف با یکدیگر نشستی را ترتیب می‌دهند، در ابتدا مجبور هستند یکدیگر را تائید هویت کنند و یک

کلید نشست ایجاد نمایند. در این خصوص، پروتکلهای احراز هویت متفاوتی وجود دارد، شامل: روش مبتنی بر یک شخص ثالث و معتمد، روش دیفی ـ هلمن، روش Kerberos و روش رمزنگاری کلید عمومی.

امنیت نامه‌های الکترونیکی را می‌توان براساس ترکیب روشهایی که در این فصل معرفی شدند، تأمین کرد. به عنوان مثال در PGP ابتدا پیامها فشرده شده و سپس توسط روش IDEA رمز می‌شوند. کلید رمز IDEA، بکمک کلید عمومی گیرندهٔ پیام، رمز می‌شود و به همراه پیام ارسال می‌گردد. بعلاوه برای بررسی صحت پیامها، از درون آن یک رشتهٔ Hash استخراج شده و امضاء می‌شود.

امنیت وب نیز یکی از عناوین مهم در امنیت شبکه است که با مقولهٔ «نامگذاری امن» آغاز می‌شود. DNSsec روشی است که نامها خودشان صحت خود را گواهی کنند و بدین ترتیب از حملهٔ DNSspoofing پیشگیری می‌شود. بسیاری از وب‌سایتهای تجارت الکترونیکی برای برقراری نشستهای امن و احراز هویت شده بین مشتری و سرویس دهنده، از SSL بهره می‌گیرند. روشهای متنوعی نیز برای برخورد مطمئن با کدهای متحرک (نظیر اسکریپتها و اکتیواکس) ابداع شده است.

اینترنت موارد بسیار متعددی را بوجود آورده که تکنولوژی و سیاستهای عمومی را روبروی هم قرار داده است. برخی از موضوعات در این مقوله عبارتند از «حریم خصوصی افراد»، «آزادی بیان» و «مالکیت معنوی آثار».

مسائل

۱. رمز قطعه کد تک‌حرفی (Monoalphabetic) زیر را بشکنید. متن اصلی فقط از حروف الفباء تشکیل شده و یک قطعهٔ ادبی از آثار مشهور لوئیس کارول است.

sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

۲. رمز قطعه کد زیر را که به صورت جایگشت ستونی رمزنگاری شده، بشکنید. این متن از یک کتاب معمولی رشتهٔ کامپیوتر انتخاب شده و طبعاً کلمهٔ computer محتمل‌ترین کلمهٔ درون آن است. متن اصلی فقط از حروف الفباء انگلیسی تشکیل شده و فاصلهٔ خالی درون آن نیست. فقط برای سادگی در خواندن، متن رمز شده به صورت دسته‌های پنج حرفی نشان داده شده است (فواصل خالی را در حین محاسبات خود حذف نمائید.)

aauan cvlre rurnn dltme aeepb ytust iceat npmcy iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lbolr otoex

۳. یک رشته بیت One-Time Pad ۷۷بیتی برای متن رمز شدهٔ شکل ۸-۴ پیدا کنید تا متن "Donald Duck" را تولید نماید.

٤. رمزنگاری کوانتومی به یک تفنگ فوتونی نیاز دارد تا بتواند در صورت نیاز، یک تک فوتون حامل بیت ۱ تولید نماید. محاسبه نمائید که بر روی یک فیبرنوری صد گیگاهرتز یک تک بیت حامل چه تعداد فوتون است. فرض کنید که طول یک فوتون معادل طول موج آنست که در این مسئله یک میکرون فرض شده است. سرعت نور در یک فیبرنوری را ۲۰ سانتی‌متر در نانو ثانیه (20Cm/nsec) در نظر بگیرید.

۵. وقتی از رمزنگاری کوانتومی استفاده می‌شود اگر ترودی بتواند فوتونهای نوری را دریافت و بازتولید نماید برخی از بیتها را اشتباه دریافت خواهد کرد و همچنین خطاهایی را در رشته بیت One-Time Pad متعلق به باب بوجود خواهد آورد. بطور متوسط چه نسبتی از رشته بیت باب خراب خواهد شد؟

۶. یکی از اصول پایهٔ رمزنگاری بیان می‌کند که تمام پیامها باید افزونگی داشته باشد. ولیکن از طرفی با این موضوع آشنا شدیم که وجود افزونگی به اخلالگر کمک می‌کند تا صحت حدس خود در مورد کلید رمز را بررسی نماید. حال به دو نوع افزونگی زیر دقت نمائید: اول آن که در n بیت از متن اصلی، یک الگوی شناخته شده قرار داده شود. دوم آن که در n بیت نهایی پیام، یک رشتهٔ Hash (استخراج شده از پیام)، قرار داده شود. آیا این دو رویکرد از دیدگاه امنیت، معادل و یکسان هستند؟ پاسخ خود را تشریح کنید.

۷. در شکل ۸-۶ (سمت راست) P-Boxها و S-boxها به صورت متناوب و یک در میان قرار گرفته‌اند. اگرچه این ساختار به ظاهر خوب و مناسب به نظر می‌رسد، آیا برای تضمین امنیت بیشتر بهتر نیست که ابتدا تمام P-Boxها و سپس تمام S-boxها قرار بگیرند؟

۸. حمله‌ای را بر علیه سیستم DES ترتیب بدهید با این دانش قبلی که متن رمز شده صرفاً از حروف الفبای انگلیسی بزرگ، فاصلهٔ خالی، کاما، نقطهٔ اعشار، سمی کالون، و کاراکترهای Line Feed و Carriage Return تشکیل شده است. هیچ چیزی در مورد بیتهای توازن (Parity Bits) در متن اصلی مشخص نیست.

۹. در این فصل محاسبه کردیم برای شکستن رمز استاندارد AES با کلید ۱۲۸ بیتی، یک ماشین رمزشکن با یک میلیارد پردازنده که می‌تواند در هر پیکوثانیه (10^{-12}sec) یک کلید را آزمایش کند، به حدود 10^{10} سال، زمان نیاز خواهد داشت. ولیکن در ماشینهای فعلی که حداکثر می‌توانند تا حدود ۱۰۲۴ پردازنده داشته باشند، برای شکستن رمز AES باید کارآیی آنها تا 10^{15} بار بهتر شود. اگر قانون Moore (که بیان می‌کند قدرت پردازش کامپیوترها هر ۱۸ ماه دو برابر می‌شود) روند خود را ادامه بدهد، چند سال طول می‌کشد تا چنین ماشینی ساخته شود؟

۱۰. AES از کلیدهای ۲۵۶ بیتی حمایت می‌کند. در AES-256 چند کلید می‌تواند وجود داشته باشد؟ بررسی کنید که آیا می‌توانید عددی معادل با این عدد در فیزیک، شیمی یا نجوم پیدا کنید. از اینترنت برای جستجوی اعداد بسیار بزرگ بهره بگیرید و نتیجه‌گیری خود از این تحقیق را ارائه بدهید.

۱۱. فرض کنید که پیامی با استفاده از DES و در حالت زنجیره‌سازی بلوکها (Block Chaining) رمزنگاری شده است. در حین انتقال یک بیت از متن رمزشده در بلوک C_i تصادفاً از صفر به یک تبدیل شده است. در اثر این خطا چقدر از متن اصلی (پس از رمزگشایی) خراب و بلااستفاده خواهد شد؟

۱۲. حالا مجدداً سیستم DES در حالت زنجیره‌سازی بلوکها را در نظر بگیرید. فقط به جای آنکه یک بیت صفر در حین انتقال به ۱ تبدیل شود یک بیت صفر اضافی تصادفاً در لابلای متن رمز شده بعد از بلوک C_i اضافه می‌شود. در اثر این خطا (پس از رمزگشایی) چقدر از متن اصلی خراب و بلااستفاده خواهد شد؟

۱۳. روش زنجیره‌سازی بلوکها (Block Chaining) را با روش Cipher Feedback Mode برحسب تعداد عملیات رمزنگاری مورد نیاز برای انتقال یک فایل بزرگ مقایسه نمائید. کدامیک از این روشها کارآمدتر و سریعتر است و چقدر؟

۱۴. در سیستم رمزنگاری کلید عمومی RSA، کاراکترهای a با عدد ۱، b با عدد ۲، c با عدد ۳ و به همین ترتیب کدگذاری شده‌اند:

الف) اگر p=7 و q=11 باشد، پنج مقدار معتبر برای d بیابید.

ب) اگر p=13 و q=31 و d=7 باشد، e را پیدا کنید.

ج) با داشتن p=5 و q=11 و d=27، ابتدا e را یافته و سپس متن "abcdefghij" را رمز کنید.

۱۵. فرض کنید کاربری به نام ماریا متوجه می‌شود که کلید **خصوصی** RSA او یعنی (d1,n1) دقیقاً معادل با کلید

عمومی RSA از یک کاربر دیگر به نام فرانسیس با کلید (e2,n2) است. به عبارت دیگر d1=e2 و n1=n2 است. آیا ماریا باید کلیدهای عمومی و خصوصی خود را تغییر بدهد؟ پاسخ خود را تشریح کنید.

۱۶. به سیستم رمزنگاری نشان داده شده در شکل ۸-۱۵ (یعنی روش Counter Mode) دقت کنید با این تفاوت که IV را در این شکل معادل صفر در نظر بگیرید. آیا عموماً استفاده از صفر برای IV ، امنیت این سیستم رمز را به خطر می‌اندازد؟

۱۷. پروتکل امضای دیجیتالی نشان داده شده در شکل ۸-۱۸ دارای این ضعف است که اگر ماشین باب به ناگاه مختل شود (اصطلاحاً crash کند) تمام محتویات RAM از دست خواهد رفت. این مسئله منجر به بروز چه اشکالی می‌شود و او چگونه می‌تواند از بروز این اشکال پیشگیری نماید.

۱۸. در شکل ۸-۲۰ می‌بینیم که چگونه آلیس پیامی امضاء شده را برای باب می‌فرستد. اگر ترودی متن P را کلاً عوض کند باب متوجه خواهد شد. به نظر شما اگر ترودی هم P و هم امضای آن را عوض کند چه اتفاقی می‌افتد؟

۱۹. امضاهای دیجیتالی دارای یک ضعف بالقوه هستند که از تنبلی کاربران ناشی می‌شود. در معاملات تجارت الکترونیکی پیش‌نویس یک قرارداد تهیه شده و از کاربر خواسته می‌شود تا رشتهٔ SHA-1 Hash متناظر با آن قرارداد را با کلید خصوصی خود امضاء نماید. اگر کاربر بررسی نکند که آیا حقیقتاً رشتهٔ Hash که آنرا امضاء می‌کند متناظر با قرارداد مورد نظر اوست ممکن است سهواً Hash یک قرارداد دیگر را امضاء کند. فرض کنید مافیا سعی می‌کند از این اشکال سوءاستفاده کرده و پول بدست بیاورد. آنها یک وب‌سایت که کاربران برای ورود به آن مجبورند پول بپردازند ایجاد کرده و از مشتریان خود می‌خواهند که شمارهٔ کارت اعتباری خود را وارد نمایند. سپس قراردادی را برای مشتری می‌فرستند تا آنرا امضاء کند، با این نیّت که اکثر کاربران بدون بررسی آنکه آیا Hash مربوط به قرارداد متناظر و متعلق به پیام هست یا خیر، آن را امضاء می‌کنند. نشان بدهید که چگونه مافیا می‌تواند مقداری الماس از یک جواهرفروش در اینترنت بخرد و قیمت آن را بر گردن یک کاربر که کسی به او مشکوک نخواهد شد بیندازد؟

۲۰. یک کلاس ریاضی ۲۰ دانش‌آموز دارد. احتمال آن که حداقل دو دانش‌آموز در یک روز از سال به دنیا آمده باشند چقدر است؟ فرض کنید هیچکس در سال کبیسه به دنیا نیامده باشد و روزهای مختلف تولد ۳۶۵ حالت بیشتر نیست.

۲۱. در سناریوی بخش ۸-۴-۴ پس از آن که الن نزد مرلین اعتراف کرد که در خصوص رسمی شدن تام در منصب هیئت علمی دانشگاه فریبکاری کرده است، مرلین برای پیشگیری از این مسئله سعی می‌کند ضمن دیکته کردن پیامهای آتی خود از طریق یک ماشین خاص، منشی خود را نیز عوض کند. سپس مرلین به گونه‌ای برنامه‌ریزی می‌کند تا از آن به بعد پیامهای تایپ شده را پس از تایپ بدقت بررسی کند تا مطمئن شود کلمات متن بدقت و صحیح تایپ شده باشند. آیا منشی جدید هنوز هم می‌تواند از «حملهٔ روز تولد» (Birthday Attack) برای جعل پیامها بهره بگیرد و اگر می‌تواند چگونه؟ (راهنمایی: این کار ممکن است).

۲۲. به تلاش ناموفق آلیس در دریافت کلید عمومی باب در شکل ۸-۲۳ دقت نمائید. فرض کنید باب و آلیس از قبل بر روی یک کلید سرّی و مشترک توافق کرده‌اند ولی باز هم آلیس به کلید عمومی باب نیاز دارد. آیا در چنین حالتی روشی برای دریافت مطمئن این کلید وجود دارد؟ اگر وجود دارد چگونه؟

۲۳. آلیس می‌خواهد با باب به کمک کلید عمومی، تبادل اطلاعات داشته باشد. او یک اتصال با کسی که فکر می‌کند باب است برقرار می‌نماید و از طرف مقابل می‌خواهد که کلید عمومی خود را بفرستد؛ طرف مقابل نیز کلید عمومی خود را به صورت آشکار و به همراه گواهینامهٔ X.509 خود که توسط مرکز عالی CA امضاء

شده ، ارسال می‌کند. آلیس نیز یک کلید عمومی که توسط مرکز CA امضاء شده در اختیار دارد. چند مراحلی باید انجام شود تا آلیس اطمینان حاصل کند که طرف مقابل او واقعاً باب است. فرض کنید باب نیز از هویت کسی که با او در حال محاوره است مطمئن نیست. (مثلاً باب یک سرویس‌دهندهٔ عمومی است.)

۲۴. فرض کنید که یک سیستم از PKI مبتنی بر ساختار «سلسله‌مراتب مراکز CA» بهره می‌گیرد. آلیس می‌خواهد که با باب ارتباط برقرار کند و به همین دلیل پس از ایجاد ارتباط، گواهینامهٔ دیجیتالی باب را که توسط یک CA با نام X امضاء شده دریافت می‌کند. فرض کنید او هیچ چیزی در مورد مرکز X نمی‌داند. آلیس باید چه مراحلی برای بررسی هویت کسی که با او صحبت می‌کند، پشت سر بگذارد.

۲۵. آیا در یک ماشین که در پشت جعبهٔ NAT (NAT BOX) قرار گرفته می‌توان از IPsec (با استفاده از AH و در **حالت انتقال**) بهره گرفت؟

۲۶. یک مزیت استفاده از HMAC به جای RSA ، برای امضای رشته‌های SHA-1 Hash را عنوان کنید؟

۲۷. یک دلیل بیاورید که چرا دیوار آتش ممکن است برای بررسی ترافیک ورودی پیکربندی شود؛ همچنین یک دلیل بیاورید که چرا دیوار آتش ممکن است برای بررسی ترافیک خروجی پیکربندی شود؛ آیا فکر می‌کنید این بررسیها احتمال موفقیت دارد؟

۲۸. قالب بستهٔ WEP در شکل ۸-۳۱ نشان داده شده است. فرض کنید که کُد کشف خطای ۳۲بیتی در این بسته، با XOR کردن کلمات ۳۲بیتی بخش داده (Payload) محاسبه شود. همچنین فرض کنید که برای رفع مشکلات RC4، از یک روش قدرتمند مبتنی بر Stream Cipher (بخش ۸-۲-۳) استفاده شود و IV به ۱۲۸ بیت توسعه یابد. آیا راهی وجود دارد که یک اخلالگر بتواند اطلاعات را استراق سمع یا دستکاری کند بدون آن که کشف شود؟

۲۹. فرض کنید که یک سازمان برای اتصال چندین سایت از طریق اینترنت به روش امن ، از VPN بهره گرفته باشد. آیا لازم است کاربری مثل جین که می‌خواهد در درون همین سازمان با کاربری دیگر به نام مری ارتباط برقرار کند از رمزنگاری یا مکانیزمهای امنیتی استفاده کند؟

۳۰. یکی از پیامهای پروتکل ۸-۳۴ را به گونه‌ای تغییر دهید تا در مقابل حملهٔ بازتاب (Reflection Attack) نفوذناپذیر شود. تشریح کنید که این تغییر شما به چه نحو کار می‌کند.

۳۱. برای ایجاد یک کلید سرّی بین آلیس و باب از روش «مبادلهٔ کلید دیفی ـ هلمن» استفاده شده است. آلیس آیتمهای (۱۹۱ و ۳ و ۷۱۹) را برای باب می‌فرستد. باب با (۵۴۳) پاسخ می‌دهد. عدد سرّی و محرمانهٔ آلیس یعنی x معادل ۱۶ است. کلید سرّی و مشترک چیست؟

۳۲. اگر آلیس و باب هیچگاه یکدیگر را ملاقات نکرده و هیچ گواهینامهٔ دیجیتالی یا کلید سرّی در اختیار نداشته باشند، می‌توانند توسط الگوریتم دیفی ـ هلمن یک کلید مشترک و سرّی ایجاد نمایند. تشریح کنید که در این الگوریتم چرا مقابله با حملهٔ نوع man-in-the-middle بسیار دشوار است؟

۳۳. در پروتکل شکل ۸-۳۹ چرا مشخصهٔ A به صورت آشکار (رمز نشده) بهمراه کلید رمزشدهٔ نشست ارسال می‌شود؟

۳۴. در پروتکل شکل ۸-۳۹ اشاره کردیم که اگر هر قطعهٔ متن پیام با ۳۲ بیت صفر شروع شود یک تهدید امنیتی به وجود خواهد آمد. فرض کنید که هر پیام با یک شمارهٔ تصادفی که به ازای هر کاربر تولید می‌شود و یک کلید سرّی که فقط برای کاربر و KDC مشخص است، شروع شود. آیا این ساختار مشکل حمله از طریق «متون شناخته شده» (Known Plaintext) را حل می‌کند؟ چرا؟

۳۵. در پروتکل «نیدهام ـ شرودر»، آلیس دو رشتهٔ چالش R_A و R_{A2} تولید می‌کند. این کار بی‌مورد و زائد به نظر

می‌رسد. آیا یکی از آنها کافی نیست؟

۳۶. فرض کنید سازمانی برای احراز هویت کاربران از روش Kerberos استفاده کرده باشد. از دیدگاه «توانایی دسترسی به سرویسهای شبکه» و «امنیت»، چه اتفاقی می‌افتد اگر AS و TGS از کار بیفتند؟

۳۷. در پروتکل احراز هویت با کلید عمومی (شکل ۸-۴۳)، در پیام ۷، R_B با کلید K_S رمزنگاری شده است. آیا این رمزنگاری لازم بوده و آیا می‌تواند به صورت رمز نشده و آشکار برگشت داده شود؟ پاسخ خود را شرح بدهید.

۳۸. ترمینالهای فروش که از کارتهای اعتباری مغناطیسی و کدهای PIN استفاده می‌کنند یک اشکال جدی دارند: یک فروشندهٔ متقلب می‌تواند دستگاه کارت‌خوان خود را به گونه‌ای دستکاری کند تا بتواند اطلاعات درون کارت و همچنین PIN افراد را بدست آورده و در جایی ذخیره نماید و از آنها برای معاملات آتی خود سوءاستفاده کند. نسل آیندهٔ ترمینالهای فروش از کارتهایی استفاده می‌کنند که بر روی آنها یک CPU کامل، صفحهٔ کلید و یک نمایشگر کوچک قرار گرفته است. پروتکلی برای این سیستم ابداع کنید تا هیچ فروشندهٔ بدخواهی نتواند آن را بشکند.

۳۹. دو دلیل بیاورید که چرا PGP پیامها را فشرده می‌کند.

۴۰. با فرض آن که همه در اینترنت از PGP استفاده کرده باشند، آیا یک پیام PGP می‌تواند برای هر آدرس دلخواه در اینترنت ارسال شود و به راحتی توسط تمام گیرندگان آن رمزگشایی گردد؟ پاسخ خود را تشریح کنید.

۴۱. در حمله‌ای که در شکل ۸-۴۷ نشان داده شده یک مرحله باقیمانده است. البته این مرحله برای موفقیت در DNS spoofing الزامی نیست ولیکن انجام چنین مرحله‌ای احتمال آن که پس از انجام عملیات شکّ دیگران برانگیخته شود را کاهش خواهد داد. به نظر شما این مرحلهٔ باقیمانده چیست؟

۴۲. پیشنهاد شده که برای خنثی کردن DNS spoofing که با استفاده از تخمین و تعیین ID عملی می‌شود، به جای استفاده از شمارنده‌ای که IDهای متوالی تولید می‌کند، از ID تصادفی استفاده شود. جنبه‌های امنیتی این روش را تشریح کنید.

۴۳. در پروتکل انتقال دادهٔ SSL، دو عدد (nonce) و یک شاه‌کلید اولیه بکار رفته است. این اعداد (nonce) چه مقادیری و چه کاربردی دارند؟

۴۴. تصویر شکل ۸-۵۵-ب در برگیرندهٔ متن اَسکی پنج نمایشنامه از شکسپیر است. آیا می‌توان به جای متن، یک قطعه موزیک را درون این تصویر از گورخرها پنهان کرد؟ اگر بله این کار چگونه ممکن است و چقدر موزیک می‌توان در آن ذخیره کرد؟ اگر جواب منفی است چرا؟

۴۵. آلیس یکی از کاربران دائمی یک «نامه‌پراکن ناشناس» از نوع Anonymous Remailer 1 بود. او پیامهای زیادی برای گروه خبری مورد علاقه‌اش alt.fanclub.alice می‌فرستاد و همه می‌دانستند که این پیامها از طرف آلیس می‌آید چرا که همهٔ پیامهایش با یک اسم مستعار مشابه می‌رسیدند. با فرض آن که سرویس‌دهندهٔ نامه‌پراکن (Remailer) به درستی کار کرده باشد، ترودی نمی‌توانسته خودش را به جای آلیس جا بزند. پس از آن که این سرویس‌دهندهٔ نامه‌پراکن (Remailer) از کار افتاد، آلیس به یک سرویس‌دهندهٔ Cypherpunk Remailer تغییر سرویس‌دهنده داد و با این سرویس‌دهنده ارسالهای خود به گروه خبری را از سر گرفت. راهی ابداع کنید تا ترودی نتواند خود را به جای آلیس جا زده و به جای او پیامهایی را برای گروه خبری ارسال نماید.

۴۶. در اینترنت به دنبال یک مورد جالب در خصوص موضوع «حفظ حریم خصوصی افراد» (Privacy) بگردید

و یک گزارش یک صفحه‌ای تهیه کنید.

۴۷. در اینترنت بدنبال چند مورد قضایی در خصوص «مالکیت حقوق معنوی آثار» (Copyright) جستجو کرده و خلاصهٔ یافته‌های خود را در یک صفحه، گزارش نمائید.

۴۸. برنامه‌ای بنویسید که ورودی خود را با XOR کردن آن با یک کلید Keystream، رمز نماید. یک مولد اعداد تصادفی مناسب بنویسید (یا پیدا کنید) تا بتوانید Keystream تولید نمائید. برنامهٔ شما باید همانند یک فیلتر عمل کرده و متن اصلی را از طریق ورودی استاندارد دریافت نماید و نتیجهٔ رمز شده را به خروجی استاندارد بفرستد؛ (و بالعکس برای رمزگشایی همینطور عمل کند). برنامه فقط باید یک پارامتر از ورودی دریافت کند که آن هم کلیدی است که از آن بعنوان نقطهٔ شروع مولد عدد تصادفی (Seed) استفاده می‌شود.

۴۹. پروسیجری بنویسید که رشتهٔ SHA-1 Hash متناظر با یک بلوک داده را محاسبه نماید. این پروسیجر باید دو پارامتر داشته باشد: یک اشاره‌گر به بافر ورودی و یک اشاره‌گر به یک بافر بیست بایتی خروجی. برای آن که شرح دقیق و جزئیات SHA-1 را بررسی کنید، در اینترنت FIPS 180-1 را جستجو نمایید که شامل شرح دقیق این استاندارد است.